

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

THIS PAGE BLANK (USPTO)

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau



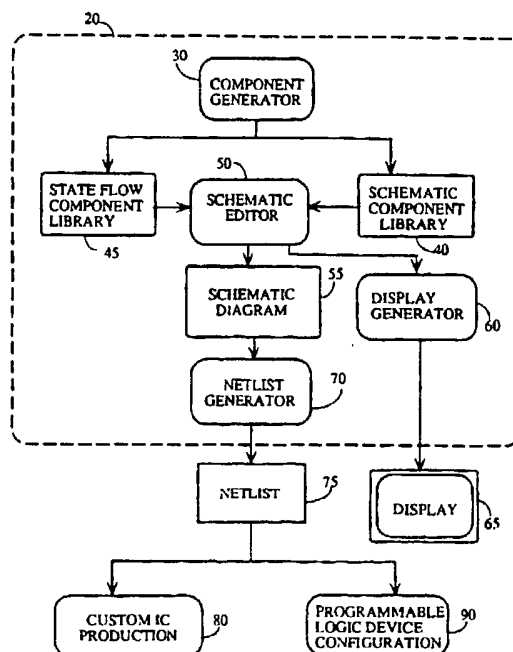
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁵ : G06F 15/60	A1	(11) International Publication Number: WO 94/15311 (43) International Publication Date: 7 July 1994 (07.07.94)
(21) International Application Number: PCT/US93/12683 (22) International Filing Date: 28 December 1993 (28.12.93) (30) Priority Data: 997,993 28 December 1992 (28.12.92) US 08/100,521 30 July 1993 (30.07.93) US (71) Applicant: XILINX, INC. [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US). (72) Inventor: DUNCAN, Robert, G.; 24 Springpoint Road, Castroville, CA 95012 (US). (74) Agents: YOUNG, Edel, M. et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).		(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD FOR ENTERING STATE FLOW DIAGRAMS USING SCHEMATIC EDITOR PROGRAMS

(57) Abstract

A system and method for entering a circuit design into a computer using a schematic capture package (20). The schematic capture package (20) is modified to include a library of state flow components (45) represented by symbols which can be connected to produce a desired representation of the circuit design. The system allows a circuit design to be displayed on a video terminal (65) using both state flow diagram and the schematic diagram symbols (55), with terminals of the state flow symbols connecting to terminals of the schematic symbols. The state schematic diagram includes schematic symbols to generate a netlist (75) representing the combined circuit.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

METHOD FOR ENTERING STATE FLOW DIAGRAMS USING
SCHEMATIC EDITOR PROGRAMS

Robert G. Duncan

6 FIELD OF THE INVENTION

7 The present invention relates to computer-aided design
8 methods for programming programmable integrated circuits, and
9 more particularly to a system for entering a circuit design
10 which includes both schematic diagrams and state flow
11 diagrams into a computer.

13 BACKGROUND OF THE INVENTION14 State Machines

15 State machines are sequential logic systems whose output
16 signals are a function of previous and present input signals,
17 in contrast to combinatorial logic systems whose output
18 signals are a function of present input signals alone. State
19 machines typically include one or more storage elements and
20 occupy a plurality of states which are determined by input
21 signals and the contents of the one or more storage elements.
22 State machines "move" sequentially between the states (that
23 is, one state becomes inactive and another state becomes
24 active) in response to the input signals and transition rules
25 established by combinatorial logic, which defines a logic
26 "path" between the states. State machines are typically
27 incorporated into circuits which also include other
28 combinatorial logic and sequential logic circuitry, and
29 frequently serve as controllers.

30 Figure 1 shows a bubble flow diagram of a state machine
31 which includes seven states, STATE 1 to STATE 7. The state
32 machine receives input signals A through E. As indicated in
33 Fig. 1, the state machine remains in STATE 1 until one of two
34 transition rules is satisfied: (1) if input signal A is
35 high, B is high, and C is low, the state machine moves into
36 STATE 4; (2) if input signal A is high, B is low and C is
37 high, the state machine moves into STATE 2. Similarly, the
38 state machine remains in STATE 4 until input signal A is
39 high, B is high and C is low. From STATE 2, the state

1 machine enters STATE 3 if input signal D is high, or enters
2 STATE 4 if input signal D is low. The other transition rules
3 resulting in movements between states are also indicated.

4 State machines are commonly implemented in programmable
5 logic devices. Programmable logic devices include
6 programmable array logic devices (PALs) and field
7 programmable gate arrays (FPGAs).

8 The state of a typical state machine is defined by the
9 states (high or low output signals) of a set of flip-flops
10 which are part of the state machine. In PALs, which have
11 wide combinatorial logic circuitry but relatively few flip-
12 flops, state machines are typically implemented using a
13 highly encoded system wherein the output signals of all of
14 the flip-flops define the active state of the state machine.
15 For example, four flip-flops are used to define 16 separate
16 states of a 16-state machine. Another scheme, call one-hot
17 encoding (OHE), is frequently used to implement state
18 machines in FPGAs, which contain numerous flip-flops but
19 limited fan-in capability. In OHE, each state is represented
20 by a single flip-flop. Thus, 16 flips are used to define 16
21 separate states. As suggested by the name OHE, only one
22 flip-flop associated with the state machine need be in a
23 logical "1" (hot) state to represent the active state of the
24 state machine. OHE requires a larger number of flip-flops
25 than are required in the highly encoded systems implemented
26 in PAL devices, but offers higher performance (speed) because
27 there are fewer levels of logic between the flip-flops.

29 Design Entry Methods

30 Circuit design engineers typically employ one or more
31 computer-aided engineering (CAE) software programs to
32 generate and test their circuit designs. Circuit designs are
33 entered into computer memory using one of several design
34 entry methods. Typical CAE programs include some form of
35 netlist generator for producing netlists from circuit
36 designs. A netlist is a listing of the components and
37 interconnections of the circuit design which is usable either

1 to produce a custom IC or to configure a programmable logic
2 device.

3 The design entry methods used for entering circuit
4 designs into computer memory include equations, truth tables,
5 waveforms (timing diagrams), schematic diagrams, state (flow)
6 diagrams and hardware description languages. Equations are
7 groups of Boolean logic equations which describe the logic
8 functions of a circuit. Truth tables are tabulations
9 indicating the relation of all output logic levels of a
10 circuit to all possible combinations of input logic levels.
11 Waveforms or timing diagrams are graphical representations of
12 the relationships between input signals and corresponding
13 output signals of a circuit. Schematic diagrams are
14 descriptions of the physical components and interconnections
15 of a circuit. Finally, flow diagrams ("flowcharts") are
16 functional descriptions of the logic functions of a circuit.

18 Schematic Diagrams

19 Schematic diagrams are considered by many circuit
20 designers to be the most intuitive method of circuit design
21 entry. Schematic diagram software programs (referred to
22 herein as "schematic capture packages") allow a circuit
23 design to be entered into computer memory in the form of
24 schematic diagram comprising circuit component symbols
25 connected by signal paths (interconnection lines). A circuit
26 designer can "read" a schematic diagram displayed on a video
27 screen and understand the interrelationships of the circuit
28 components without a specialized knowledge of the schematic
29 capture package.

30 As shown in Fig. 2, a schematic capture package 1000
31 typically includes several software tools including a
32 component generator 1010, a schematic editor 1020, a display
33 generator 1030 and a netlist generator 1040. A netlist is a
34 computer memory file including a list of components (logic
35 gates, flip flops, etc.) and interconnections between the
36 components which represent a circuit design. The component
37 generator 1010 allows a user to define and store schematic

1 components in a schematic component library 1015. A
2 schematic component is comprised of two parts: a schematic
3 symbol which is displayed on a video monitor, and an
4 underlying circuit design which defines the function of the
5 schematic component. Schematic components are recognized by
6 the schematic capture package as representing the functions
7 of their underlying circuit designs. After a schematic
8 component library 1015 is generated, the schematic editor
9 1020 is used to copy schematic components from the library
10 and to connect the schematic components to form a circuit
11 design which is stored in a schematic diagram file 1025.
12 During the process of forming a schematic diagram, the
13 display generator 1030 reads the schematic diagram file 1025
14 and generates a schematic diagram on a video display 1035.
15 After completion of the schematic diagram, the netlist
16 generator 1040 reads the schematic diagram file 1025 and
17 converts the circuit design into a netlist 1050. Netlist
18 1040 differs from schematic diagram 1025 in form. The
19 schematic diagram is a graphical illustration easily
20 recognized by a user, while the netlist is a list of
21 components and the lines which interconnect them, and is
22 easily manipulated by a computer.

23 Figs. 3(a) to 3(d) are provided to illustrate how the
24 component generator 1010 (Fig. 2) is used to simplify the
25 production of schematic diagrams. Fig. 3(a) shows a low-
26 level schematic diagram of a multiplexer (MUX). The low-
27 level schematic diagram is drawn at the "transistor level";
28 that is, the schematic diagram is comprised of individual
29 logic element symbols representing discrete transistors.
30 Fig. 3(b) shows a middle-level schematic diagram of the MUX.
31 The middle-level diagram is drawn at the "gate level"; that
32 is, related groups of transistors are represented by symbols
33 commonly recognized by circuit designers as AND gates 1110
34 and 1120, and an OR gate 1130. Note that the input signals
35 IN1 and IN2, the select signal SEL and the output signal OUT
36 of the low-level diagram of Fig. 3(a) are identified in the
37 middle-level diagram of Fig. 3(b). Fig. 3(c) shows a high-

1 level symbol 1150 representing the MUX of Figs. 11(a) and
2 11(b). The high-level symbol 1150 represents the MUX as a
3 trapezoid having input terminals IN1, IN2, SEL and OUT. When
4 the high-level symbol 1150 is entered into a schematic
5 diagram, as shown in Fig. 3(d), the high-level symbol 1150
6 will be recognized by a schematic capture package as being
7 comprised of the circuit shown in Fig. 3(a).

8 The process of defining high-level components in terms
9 of lower-level schematic diagrams using the component
10 generator 1010 (Fig. 2) is known as "hierarchy". Hierarchy
11 greatly reduces the time necessary to generate a circuit
12 diagram because it allows the circuit designer a short-hand
13 method of entering commonly-known or repeatedly-used circuit
14 structures. Further, the higher-level symbols, such as the
15 MUX symbol of Fig. 3(c), are well known; therefore, circuit
16 designers need not design at the transistor level, which is
17 time-consuming and tedious. Schematic capture programs
18 typically provide a library of commonly-used schematic
19 components. In addition, schematic component libraries are
20 provided by programmable logic device manufacturers. Also, a
21 library may be supplemented by schematic components generated
22 by the user.

23 The schematic editor 1020 of a schematic capture package
24 1000 is a user's schematic diagram production tool. With the
25 schematic editor 1020, the user accesses the schematic
26 component library 1040 and copies the schematic components
27 into a schematic diagram file 1025. The user draws signal
28 lines between the input and output terminals of the schematic
29 components to produce a representation of a desired circuit
30 design. As the user enters the schematic components and
31 interconnecting lines into computer memory, the computer
32 generates a graphic representation of the schematic diagram
33 on a video display terminal 1035 in which the symbols
34 representing the schematic components are shown along with
35 interconnection lines representing conductive signal paths
36 for transmitting signals between the schematic components.
37 The user "reads" the displayed schematic diagram by looking

1 at the signal paths and schematic components shown on
2 display terminal 1035. The user then modifies or adjusts the
3 circuit design by modifying the schematic diagram file 1025
4 based on feedback provided from display terminal 1035.

5 Finally, after a user is satisfied with having entered
6 a circuit design into the computer using the schematic editor
7 1020, the netlist generator 1030 reads the schematic diagram
8 file 1025 and converts the circuit design into a netlist
9 1050.

10 A presently available schematic capture package is
11 Workview™, which is available from ViewLogic Systems, Inc.,
12 having a principle place of business at Marlboro, MA.

13 A disadvantage of prior art schematic capture packages
14 is that when a circuit design includes a state machine, the
15 schematic diagram of the state machine, which must be
16 entered by the user, typically fails to intuitively present
17 the logic defining the state machine.

19 Flow Diagrams

20 Flow diagrams (flowcharts) are considered by many
21 circuit designers to be the most intuitive design entry
22 method for entering state machine descriptions into computer
23 memory. Flow diagrams are graphical and/or text based
24 descriptions which illustrate the paths between the states of
25 a state machine.

26 Fig. 4 shows an example of a flow diagram of a four-
27 state machine. In Fig. 4, the rounded rectangles 41 through
28 44 indicate individual states and the diamonds 45, 46
29 indicate decision branches controlled by test signals (not
30 shown). Linking the states and decision branches are arrows
31 51 through 56 indicating flow paths. The state machine moves
32 from state to state along the flow paths as directed by the
33 decision branches of the flow diagram.

34 Kobayashi et al. U.S. Patent No. 4,922,432 teaches a CAE
35 flow diagram program which allows a circuit designer to enter
36 a state machine as a flow diagram comprised of blocks
37 representing states, diamonds representing decision branches,

1 and arrows indicating the logic paths linking the states and
2 decision branches. Associated with one or more of the states
3 is a textual description of logic controlled by that state.
4 In addition, each decision branch identifies a logic equation
5 stored in memory.

6 However, Kobayashi et al. does not display the
7 combinatorial and sequential logic circuitry which is
8 associated with the state machine. That is, although a user
9 can understand the logic of the state machine, the user
10 cannot easily understand the relationships between the state
11 machine and the related circuit design. Specifically, for
12 applications in which the state machine is a small part of
13 the overall circuit design, it can be difficult for the
14 circuit designer to comprehend the overall circuit design.

15 Another disadvantage of the above-mentioned flow diagram
16 program is that it requires a user to learn a new design
17 entry format which is incompatible with a schematic capture
18 package. Circuit designers who are familiar with schematic
19 capture packages may find the new format difficult to learn.
20 Further, because the flow diagram program is incompatible
21 with schematic capture packages, a user must purchase both
22 the flow diagram program and a schematic capture package in
23 order to enter circuit designs in both the schematic diagram
24 format and the flow diagram format. These programs are
25 typically expensive.

26

27 SUMMARY OF THE INVENTION

28 The present invention overcomes the disadvantages of the
29 prior art by providing a method and a system wherein a
30 schematic capture package includes state components having
31 associated symbols which can be arranged into a state flow
32 diagram. The state flow diagram can be located on the same
33 graphic display as a circuit diagram, and connected with
34 schematic symbols of the schematic diagram. This improves
35 the ability of a user to enter and edit a circuit design for
36 use in producing a custom IC or to configure a programmable
37 logic device because state machine information can be entered

1 in a flow diagram format.

2 In one embodiment of the invention, the state flow
3 components include START, STATE, IF and JOIN components which
4 are connected by a flow bus. The START and STATE components
5 include ACTIVE output signal lines which can be connected to
6 schematic components to provide information regarding the
7 active state of a state machine. The IF component includes a
8 TEST input line which can receive a signal from a schematic
9 component to direct the flow of the state machine along one
10 of two flow bus segments. The JOIN component combines
11 signals from two or more flow bus segments. In addition to
12 the above-mentioned state flow components, additional higher-
13 level components can be defined by the user.

14 In another embodiment, the components include START,
15 STATE, IF, JOIN2, JOIN3, PAUSE, FOR, CALL, RETURN, STOP, and
16 SPAWN. The PAUSE component can replace a specified number of
17 STATE symbols. The user enters the number in a space
18 provided on the symbol. The PAUSE symbol thus saves chip
19 hardware. The FOR symbol is used to create iterative FOR
20 loops and also includes a cycle number. The CALL symbol
21 calls a subroutine and the RETURN symbol returns from the
22 call. The SPAWN symbol provides a splitting point for a
23 state machine with multiple active branches. When a SPAWN
24 symbol is used, the system no longer has the characteristic
25 of having only one state active at one time. With a state
26 machine having multiple active branches, care must be taken
27 to assure that the separate branches do not corrupt the
28 integrity of each other.

29 In yet another embodiment, the components further
30 include ACTIVATE and TAP symbols. The ACTIVATE symbol
31 induces a state to become active in response to an event
32 external to the state diagram. Like the SPAWN symbol, the
33 ACTIVATE symbol may cause a state machine to have more than
34 one active state at one time. The complement to the ACTIVATE
35 symbol is the TAP symbol. The TAP symbol generates a signal
36 for controlling a part of the system external to the state
37 diagram.

1 In other embodiments additional symbols or combinations
2 of symbols are provided. If it is desired to provide
3 additional state machine symbols, it is not necessary to
4 write new software except to the extent of creating a new
5 symbol and its underlying circuit, and adding this to the
6 existing library.

7 As another feature of the invention, a set of components
8 having invalid state elimination logic is included. These
9 symbols force certain states to become inactive in response
10 to particular states becoming active, thus overcoming invalid
11 states which may result from power glitches or other
12 unexpected phenomena.

13 A system incorporating the present invention includes a
14 schematic editor program which accesses a library containing
15 one or more state flow symbols, each having an active output
16 terminal, and one or more schematic symbols. The state flow
17 symbols can be arranged to form a state flow diagram whose
18 logic is easily comprehended by a user. The input or output
19 terminal of at least one state flow symbol is connected to at
20 least one schematic symbol to produce a single graphical
21 representation of a circuit design.

23 BRIEF DESCRIPTION OF THE DRAWINGS

24 These and other features, aspects, and advantages of the
25 present invention will become better understood with regard
26 to the following description, appended claims and
27 accompanying drawings.

28 Fig. 1 shows a bubble diagram illustrating a state
29 machine.

30 Figs. 2 shows the main portions of a prior art schematic
31 capture package.

32 Fig. 3a through 3d various levels of schematic diagrams.

33 Fig. 4 shows a logic flow diagram of a state machine.

34 Fig. 5 shows a block diagram of a system incorporating
35 the present invention.

36 Fig. 6a shows a symbol for a START component in
37 accordance with the present invention.

1 Fig. 6b shows the equivalent circuit for the START
2 symbol of Fig. 6a using schematic components provided in the
3 X-BLOX™ symbol library available from Xilinx, Inc.

4 Figs. 7a and 7b show a symbol and equivalent circuit for
5 a STATE component in accordance with the present invention;

6 Figs. 8a and Fig. 8b show a symbol and equivalent
7 circuit for a JOIN component in accordance with the present
8 invention; Fig. 8c shows another JOIN component.

9 Figs. 9a and 9b show a symbol and equivalent circuit for
10 an IF component.

11 Fig. 10a shows an example of a UART design including a
12 state flow diagram and a schematic diagram.

13 Fig. 10b shows an equivalent schematic diagram for the
14 display of Fig. 10a.

15 Figs. 11a and 11b show a symbol and equivalent circuit
16 for a PAUSE component.

17 Fig. 11c shows use of the PAUSE component in the UART
18 example of Fig. 10a.

19 Figs. 12a and 12b show a symbol and equivalent circuit
20 for a FOR component in accordance with the present invention.

21 Figs. 12c and 12d show a symbol and equivalent circuit
22 for a WHILE component.

23 Figs. 13a and 13b show a symbol and equivalent circuit
24 for a CALL component.

25 Figs. 14a and 14b show a symbol and equivalent circuit
26 for a RETURN component.

27 Fig. 15 shows a UART transmitter used in a subroutine
28 with the CALL and RETURN symbols.

29 Figs. 16a and 16b show a symbol and equivalent circuit
30 for a STOP component.

31 Figs. 17a and 17b show a symbol and equivalent circuit
32 for a SPAWN component.

33 Fig. 18 shows a UART transmitter-receiver which uses the
34 SPAWN symbol to provide a separate branch to transmit data in
35 parallel with receiving data.

36 Figs. 19a through 19d show symbols and circuits for the
37 ACTIVATE and TAP components.

1 Fig. 20 shows an example data compression transfer
2 function which can use a state machine having multiple active
3 branches.

4 Fig. 21 shows a state machine solution to the problem of
5 Fig. 20 in which multiple active branches are used.

6 Figs. 22a through 22h show simplified circuit diagrams
7 of START,
8 STATE, JOIN2, and IF components which do and do not have
9 logic for eliminating invalid states.

10 Fig. 23 shows a portion of a state flow diagram using
11 components which have logic for eliminating invalid states.

12 Figs. 24a and 24b show a symbol and equivalent circuit
13 for a comparator IF component.

14 Figs. 25a through 25g show symbols and circuits for a
15 coded CALL or CCALL component, a 2-BUS JOIN2 component and a
16 SUBROUTINE component which are used together.

17 Fig. 26 shows a recursive state machine using the CCALL
18 component, the 2-BUS JOIN2 component and the SUBROUTINE
19 component.

20 Figs 27a and 27b show a VECTOR1 component and its
21 equivalent circuit. Fig. 27c shows a simplified circuit
22 diagram of the VECTOR1 component used to connect a STATE3
23 component with a STATE4 component.

24 Figs. 27d and 27e show a VECTOR3 component and its
25 equivalent circuit.

26 Figs. 28a and 28b show a FORR component and its circuit.

27 Figs. 29a and 29b show the related FORI component and its
28 circuit.

29

31 DETAILED DESCRIPTION OF THE DRAWINGS

32 In the following description of the present invention a
33 "circuit design" includes both state flow components and
34 schematic components. State flow components include state
35 components and other components indicating decision branches
36 and flow control. Schematic components include
37 combinatorial circuits and general purpose sequential logic

1 circuits, including counters, registers and flip-flops.

3 System Overview

4 In accordance with the present invention, a system and
5 method for entering circuit designs into a computer is
6 disclosed which allows both schematic diagram symbols and
7 state flow diagram symbols to be displayed together on a
8 computer video terminal and interpreted together by a
9 schematic capture program. In one embodiment of the
10 present invention, a commercially-available schematic capture
11 package is modified to include a library of state flow
12 components. Like a schematic component, a state flow
13 component includes two parts: a state flow diagram symbol
14 which is displayed on a video monitor, and an underlying
15 circuit design which defines the function of the state flow
16 component. The state flow diagram symbols can be arranged on
17 a video terminal to represent a state flow diagram. At least
18 one of the state flow diagram symbols includes input and/or
19 output terminals which can be connected to the terminals of
20 conventional schematic components. The library of state flow
21 components can be accessed by the schematic capture package
22 along with a library of schematic components, thereby
23 allowing a circuit design to be created which is displayed as
24 a combination of a schematic diagram symbols and flow diagram
25 symbols. Further, the interaction of the state flow
26 components and the schematic components are displayed using
27 typical interconnect lines (signal paths) connected between
28 the terminals of one or more state flow symbols and the
29 terminals of one or more schematic symbols.

30 An advantage of the present invention is that an
31 existing schematic capture package can translate the state
32 flow diagram to a computer usable form (for example, a
33 netlist) using the same software as is used to translate
34 other schematic symbols. Therefore, existing schematic
35 capture packages can be modified to incorporate the present
36 invention simply by adding the library of state flow
37 components, as described below.

1 Another advantage of the present invention is that the
2 circuit designer need not purchase new software, but need
3 only modify an existing schematic capture package to include
4 the library of basic state flow components described below.
5 Further, the circuit designer can expand the library of basic
6 state flow components to include higher level state flow
7 components, as discussed below.

9 One Embodiment of the Invention

10 In one embodiment, Xilinx, Inc., XC4000-series field
11 programmable gate arrays (FPGAs) (described in THE XC4000
12 DATA BOOK, (copyright 1992) published by Xilinx, Inc., of
13 2100 Logic Drive, San Jose, CA, 95124) are programmed in
14 accordance with the present invention. A feature of Xilinx
15 FPGAs which works well with the present invention is the
16 abundance of D flip-flops. As discussed in the background
17 section, when state components are represented by flip-flops
18 in the one-hot encoding (OHE) method, one flip flop is used
19 to represent each state. Therefore, state flow components
20 associated with the individual states of a state machine can
21 be implemented by a flip-flop and associated logic.

23 System Context

24 The system incorporating the present invention can be
25 implemented using Workview™ 4.1 schematic capture package,
26 Series I, including Viewdraw™-LCA Version 4.0 modified to
27 include Xilinx XACT® software which includes an X-BLOX™
28 symbol library. Workview™ is available from ViewLogic
29 Systems of Marlboro, MA. The Xilinx XACT® software and
30 X-BLOX™ library are available from Xilinx, Inc.

31 The Workview™ software package runs on any IBM-
32 compatible computer including a 386 microprocessor using DOS
33 3.0 or higher. The recommended configuration uses 16
34 megabytes of RAM to run the necessary software, and requires
35 user interfaces including a VGA display monitor, a manually
36 operated keyboard, a mouse and preferably 60 megabytes of
37 hard-disk memory.

1 X-BLOX™ is a library of symbols which simplify the entry
2 of the flow diagram, described below. The operation and
3 installation of the X-BLOX™ package into Workview™ is taught
4 in the X-BLOX™ Design Tool User Guide, available from Xilinx,
5 Inc.

6

7 Major Component Overview

8 As shown in Fig. 5, a CAE system incorporating the
9 present invention includes a library of state flow components
10 45 in a schematic capture package 20. The schematic capture
11 package 20 is comprised of a component generator 30, a
12 schematic editor 50, a display generator 60 and a netlist
13 generator 70.

14 The component generator 30 is used to create user-
15 defined schematic components and state flow components. The
16 user-defined components are stored in the schematic component
17 library 40 and the state flow component library 45. The
18 component generator 30 may be combined with the schematic
19 editor 50.

20 The schematic components library 40 is typically
21 provided as a part of a schematic capture package 20, but
22 additional schematic components can also be added by a user.
23 A source of additional schematic components is from
24 programmable logic device manufacturers. For instance, the
25 X-BLOX™ software from Xilinx, Inc., provides a library of
26 schematic components, which include a D flip-flop, a counter
27 and a register, all of which include placement and routing
28 instructions to be implemented in the XC4000-series FPGA.
29 X-BLOX™ components in the Xilinx library have features
30 related to the representation of buses which are used to
31 simplify the state flow components, as discussed below.

32 The schematic editor 50 is used to enter schematic
33 components from the schematic component library 40 and state
34 flow components from the state flow component library 45 into
35 the circuit design file 55. In addition, the schematic
36 editor 50 is used to enter bus flow lines connecting the
37 state flow components to form a flow diagram representing a

1 first portion of a circuit design. The schematic editor 50
2 is also used to enter signal lines connecting the schematic
3 components to produce a schematic diagram representing a
4 second portion of the circuit design. The schematic editor
5 50 is also used to connect state flow components to
6 schematic components.

7 The display generator 60 reads the circuit design file
8 55 and generates signals which drive a video display 65. The
9 image generated by the video display 65 provides feedback to
10 the user such that the user can modify the contents of the
11 circuit design file 55 until it contains a desired circuit
12 design.

13 After the desired circuit design is entered into the
14 circuit design file 55, the netlist generator 70 reads the
15 circuit design file 55 and converts the circuit design into a
16 netlist 75. The netlist 75 is used to produce custom ICs 80
17 or to configure programmable logic devices 90 to implement
18 the desired circuit design.

20 Basic State Flow Components

21 In one embodiment of the present invention, four basic
22 state flow components, START, STATE, IF and JOIN2, are
23 provided in a library, and can be arranged to form state flow
24 diagrams. The basic state flow components provide the
25 minimum amount of flow diagram logic necessary to implement
26 most state machines.

27 The basic state flow components are entered into a
28 library for use with the Workview™ schematic capture package
29 using methods described in the XILINX USER GUIDE AND
30 TUTORIALS, published by Xilinx, Inc. (1991). Further
31 information regarding X-BLOX™ schematic components used to
32 define the state flow components is provided in the X-BLOX™
33 Design Tool User Guide, available from Xilinx, Inc.

35 START Component

36 Fig. 6a shows a symbol for a START component. The
37 START component determines the initial state of a state

1 machine. That is, when the computer is powered up or when a
2 state machine is reset, the only active state of the state
3 machine is the START state indicated by the START component.
4 Each flow diagram incorporating the basic flow components
5 according to the present invention includes one START
6 component.

7 Fig. 6b shows the circuit for the START component. As
8 shown in Fig. 6b, the START component comprises a D flip-flop
9 210. Signal paths CLK_EN, CLOCK and RESET are connected to
10 the clock enable (CE), clock (C) and set (S) terminals of D
11 flip-flop 210. The D input terminal of D flip-flop 210 is
12 connected to ground. The Q output terminal of D flip-flop
13 210 is connected to the ACTIVE signal path, which is used to
14 connect the START component to schematic components. The
15 START component further includes a NEXT flow bus which is
16 defined using the BUS DEF and ELEMENT schematic components
17 provided by the X-BLOX™ library. The BUS DEF component
18 identifies the number of signal lines associated with the
19 NEXT flow bus. The ELEMENT components identify individual
20 signal lines "ELEM=0", "ELEM=1", "ELEM=2" and "ELEM=3", of
21 the NEXT flow bus. In this embodiment, the ELEM=0 signal
22 line is connected to receive the CLOCK signal, the ELEM=1
23 signal line is connected to receive the CLK_EN signal, the
24 ELEM=2 signal line is connected to receive the RESET signal,
25 and the ELEM=3 signal line is connected to receive the ACTIVE
26 signal generated from the Q output terminal of D flip-flop
27 210.

28 In operation, a START component transmits a high output
29 signal on the ACTIVE signal line when the RESET signal line
30 is high. The high ACTIVE signal is also applied to the
31 ELEM=3 signal line of the NEXT flow bus. When RESET goes
32 low, the Q output signal remains high for the remainder of
33 the clock cycle, after which time the Q output signal goes
34 low. Note that because the D input terminal of D flip-flop
35 210 is connected to ground, a START component can only
36 generate a high Q output to the ACTIVE signal line upon reset
37 of the system.

2 STATE Component

3 Figs. 7a and 7b show the symbol and circuit,
4 respectively, for a STATE component. A STATE component
5 represents a finite state of a state machine. STATE
6 components are connected to the flow bus of a state flow
7 diagram and generate a high ACTIVE signal when activated by
8 another state flow component.

9 Similar to the START component, the STATE component
10 comprises a D flip-flop 310 connected to the flow bus such
11 that the CLOCK signal transmitted on the ELEM=0 signal line
12 is connected to the C (clock) input terminal, the CLK_EN
13 signal transmitted on the ELEM=1 signal line is connected to
14 the CE (clock enable) input terminal, the RESET signal
15 transmitted on the ELEM= 2 signal line is connected to the R
16 (reset) terminal, and the ELEM=3 signal line is connected to
17 the D input terminal. Unlike the START component, the STATE
18 component includes both PREV and NEXT flow bus connections,
19 which are used to connect the STATE component between two
20 other state flow components. In addition, because the RESET
21 signal is connected to the R input terminal, the STATE
22 component outputs a low signal on the Q output terminal when
23 reset. After the reset signal goes low, when a high signal
24 is applied to the D input terminal from the ELEM=3 signal
25 line from the PREV flow bus connection, the STATE component
26 outputs a high signal on the Q output terminal upon a rising
27 CLOCK signal transition. Similar to the START component, the
28 STATE component also includes an ACTIVE signal line connected
29 to the Q output terminal of D flip-flop 310. The ACTIVE
30 signal line is connected to the ELEM=3 signal line of the
31 NEXT flow bus connection, and can also be connected to
32 schematic signal lines.

33 In operation, the STATE component applies an inactive
34 (low) signal on the ACTIVE signal line until a high signal is
35 applied to the D input terminal of D flip-flop 310 from the
36 ELEM=3 signal line of the PREV flow bus. When activated, the
37 STATE component applies a high output signal on the ACTIVE

1 signal line for one cycle of the clock, while the clock
2 enable signal CLK_EN is asserted. After the clock cycle, the
3 STATE component will either output a low cycle if the D input
4 remains low, or a high output if the D input remains high.

6 JOIN Components

7 Figs. 8a and 8c illustrate two JOIN state flow
8 components, JOIN2 and JOIN3. The JOIN components are used
9 as junction points to combine signals from two or more flow
10 buses into one flow bus. The JOIN components facilitate
11 branching of a state flow diagram by providing means for
12 joining two or more flow bus branches.

13 As shown in Fig. 8b, the two-input JOIN2 component of
14 Fig. 8a is comprised of an OR gate 410 connected to the
15 ELEM=3 signal lines of two input flow buses. In the
16 illustrated example, the input flow buses are identified as
17 PREV and RIGHT. The output of OR gate 410 is connected to
18 the ELEM=3 signal line of a NEXT flow bus. In addition, the
19 ELEM=0 signal lines, the ELEM=1 signal lines, and the ELEM=2
20 signal lines from the PREV and RIGHT flow buses are connected
21 to the ELEM=0 signal line, the ELEM=1 signal line, and the
22 ELEM=2 signal line of the NEXT flow bus, respectively. In
23 operation, a high signal on the ELEM=3 signal lines of either
24 the PREV or RIGHT flow buses is applied to the ELEM=3 signal
25 line of the NEXT flow bus.

26 Note that there is contention if two signal input
27 lines having different signals are joined together as shown
28 for the ELEM=0, ELEM=1 and ELEM=2 lines of the RIGHT and PREV
29 buses. However, in the present embodiment, these lines
30 carry the same signal on both buses; ELEM=0 carries the clock
31 signal, ELEM=1 carries the clock enable signal, and ELEM=2
32 carries the reset signal. An implementation can be provided
33 in which only the ELEM=3 line of the RIGHT bus is connected,
34 and this implementation will be preferred for designs in
35 which the clock signal or other signals are gated, producing
36 some delay, and where contention is possible. However, if
37 the ELEM=0 through ELEM=2 signal lines from the RIGHT flow

1 bus are not provided, situations may occur in which a clock
2 signal is not provided to components downstream. Therefore,
3 in the embodiment of Fig. 8b, all lines of the RIGHT flow bus
4 are connected in the JOIN2 symbol implementation. The user
5 must take care to use a symbol appropriate for the particular
6 design.

7 Fig. 8c shows a three-input JOIN3 symbol which receives
8 signals from three flow buses: PREV, LEFT and RIGHT. The
9 underlying circuitry of the three-input JOIN3 symbol is
10 similar to the two-input JOIN2 symbol shown in Figs. 8a and
11 8b with the addition of a third ELEM=3 signal line to the OR
12 gate 410. Therefore, a high signal on the ELEM=3 signal
13 lines of either the PREV, RIGHT or LEFT flow buses is applied
14 to the ELEM=3 signal line of the NEXT flow bus.

16 IF Component

17 Figs. 9a and 9b illustrate a fourth basic state flow
18 component IF. The IF component acts as a decision branch in
19 a state flow diagram. That is, the IF component passes state
20 signals along one of two flow buses in response to a control
21 signal.

22 As shown in Fig. 9a, the symbol for the IF component is
23 a diamond having a corner into which points the PREV flow
24 bus, a corner into which extends the TEST input terminal, a
25 corner from which extends the FALSE flow bus and a corner
26 from which extends the TRUE flow bus.

27 As shown in Fig. 9b, the IF component includes PREV,
28 FALSE and TRUE bus lines, and further includes a first AND
29 gate 510 and second AND gate 520. The ELEM=0, ELEM=1 and
30 ELEM=2 signal lines from the PREV flow bus are applied to the
31 ELEM=0, ELEM=1 and ELEM=2 signal lines, respectively, of both
32 the TRUE and FALSE flow buses. The first AND gate 510
33 includes an inverting input terminal 511 and a non-inverting
34 input terminal 512. A TEST signal line, which transmits a
35 high or low signal from either another state flow component
36 or a schematic component, is connected to the inverting input
37 terminal 511 of the first AND gate 510, and to the second

1 (non-inverting) input terminal 522 of the second AND gate
2 520. In addition, the ELEM=3 signal line from the PREV flow
3 bus is applied to the non-inverting input terminal 512 of the
4 first AND gate 510 and the first input terminal 521 of the
5 second AND gate 520. The output terminal of the first AND
6 gate 510 is connected to the ELEM=3 signal line of the FALSE
7 flow bus, and the output terminal of the second AND gate 520
8 is connected to the ELEM=3 signal line of the TRUE flow bus.

9 The IF component operates to direct the progression of
10 logic from a START or STATE component to another STATE
11 component. As shown in Fig. 9b, when the ELEM=3 signal line
12 from the PREV flow bus is high and the TEST signal line is
13 high, the second AND gate 520 applies a high signal to the
14 ELEM=3 signal line of the TRUE flow bus. Likewise, if the
15 ELEM=3 signal line from the PREV flow bus is high and the
16 TEST signal line is low, the first AND gate 510 applies a
17 high signal to the ELEM=3 signal line of the FALSE flow bus.
18 Signals from the ELEM=0, ELEM=1, and ELEM=2 lines of the PREV
19 flow bus are applied to the corresponding lines of both the
20 TRUE and FALSE flow buses. However, at most one of the TRUE
21 and FALSE flow buses will have a high value on its ELEM=3
22 line indicating an active state.

24 Operation Using Basic State Flow components

25 With a system including the library of state flow
26 components described above, a circuit design including both
27 schematic components arranged in a schematic diagram and
28 state flow components arranged in a flow diagram can be
29 entered into a computer memory file using a prior art
30 schematic capture package. The state flow components are
31 arranged in the form of a state flow diagram to functionally
32 describe a selected state machine, and schematic components
33 are arranged in the form of a schematic diagram adjacent to
34 the state flow diagram. State enable signals are connected
35 to selected combinatorial component input terminals by
36 drawing connective paths between the ACTIVE terminal of the
37 STATE or START component and the selected input terminal.

1 This process will be better understood by referring to the
2 following example.

3

4 UART Example

5 As shown in Fig. 10a, a state flow diagram for a 7-bit
6 UART transmitter circuit design includes one START component,
7 several STATE components, one IF component, and two JOIN2
8 components.

9 The schematic components include a data register 610, a
10 shift register 620, a D flip-flop 630, an AND gate 640, a
11 first OR gate 645 and a second OR gate 650. The data
12 register 610 includes a data input terminal D_IN, a clock
13 input terminal CLOCK and an output terminal Q_OUT. The shift
14 register 620 includes a parallel load data input terminal
15 PAR_IN, a load enable input terminal LOAD, an asynchronous
16 reset input terminal ASYNC_CTRL, a clock enable input
17 terminal CLK_EN and a clock input terminal CLOCK, and several
18 output terminals including a least significant bit serial
19 output terminal LS_OUT. The D flip-flop 630 includes D, C
20 and RD input terminals, and a Q output terminal. Note that
21 the unused input and output terminals associated with the
22 data register 610, shift register 620 and D flip-flop 630 are
23 indicated even though they are not used in this example
24 because they are provided with the corresponding schematic
25 components provided in the X-BLOX™ library. The AND gate 640
26 includes an inverting input terminal 641 and a non-inverting
27 input 642. The first OR gate 645 includes a first input
28 terminal 646 and a second input terminal 647. The second OR
29 gate 650 includes a first input terminal 651 and a second
30 input terminal 652.

31 The state flow components in this example include the
32 START component 660, two JOIN2 components 661 and 663, an IF
33 component 665 and ten STATE components 662, 664 and 666
34 through 673. The input and output terminals of each of the
35 state flow components are described above in reference to
36 Figs. 6a through 9b.

37 The state flow diagram is entered into a computer

1 memory file using X-BLOX™-defined schematic components and
2 the state flow components described above. The schematic
3 components and the state flow components are entered and
4 connected as follows. First, data register 610, shift
5 register 620, D flip-flop 630, AND gate 640, OR gate 645 and
6 OR gate 650 are accessed from the X-BLOX™ library and
7 positioned on a video display screen in the relative
8 positions shown in Fig. 10a. Next, the state flow components
9 are accessed from the state flow component library and
10 arranged as follows: the NEXT flow bus of START component
11 660 is connected to the PREV flow bus of JOIN2 component
12 661; the NEXT flow bus of JOIN2 component 661 is connected
13 to the PREV flow bus of STATE component 662; the NEXT flow
14 bus of STATE component 662 is connected to the PREV flow bus
15 of JOIN2 component 663; the NEXT flow bus of JOIN2
16 component 663 is connected to the PREV flow bus of STATE
17 component 664; the NEXT flow bus of STATE component 664 is
18 connected to the PREV flow bus of IF component 665; the TRUE
19 flow bus of IF component 665 is connected to the PREV flow
20 bus of STATE component 666; the FALSE flow bus of IF
21 component 665 is connected to the RIGHT flow bus of JOIN2
22 component 663; STATE components 667 through 673 are connected
23 in series from state component 666. Finally, the NEXT flow
24 bus of STATE component 673 is connected to the RIGHT flow bus
25 of JOIN2 component 661.

26 As the schematic components and state flow components
27 are entered, connector lines (signal paths) are connected
28 between the schematic components and between at least one
29 state flow component and at least one schematic component as
30 follows. A DATA IN data bus (which represents seven data
31 signal lines) is connected to the D_IN terminals of data
32 register 610. A LOAD input line is connected to the CLOCK
33 input terminal of data register 610 and the C input terminal
34 of D flip-flop 630. A logical high signal (VCC) is
35 connected to the D input of D flip-flop 630. A UART ENABLE
36 signal line is connected to the CLK_EN input terminal of
37 START component 660 and to the CLK_EN input terminal of

1 shift register 620. A UART CLOCK signal line is connected to
2 the CLOCK input terminal of START component 660 and to the
3 CLOCK input terminal of shift register 620. A UART RESET
4 signal line is connected to the RESET terminal of START
5 component 660 and to the ASYNC_CTRL input terminal of shift
6 register 620. The DATA IN bus, LOAD, VCC, UART ENABLE, UART
7 CLOCK and UART RESET signal lines are recognized by the
8 schematic capture package as being connected to external
9 signal sources. Internal signal lines are connected as
10 follows. A parallel data bus 611 including seven signal
11 lines is connected between the Q_OUT terminal of data
12 register 610 and the PAR_IN terminal of shift register 620.
13 In addition, schematic signal lines connect the following:
14 from the Q output terminal of D flip-flop 630 to the TEST
15 input of IF component 665, from the LS_OUT terminal of
16 shift register 620 to the non-inverting input of AND gate
17 640, from the ACTIVE output line of STATE component 662 to
18 the second input terminal of OR gate 645, from the ACTIVE
19 output line of STATE component 664 to the first input
20 terminal of OR gate 645, from the ACTIVE output line of
21 STATE component 666 to the Reset input terminal of D
22 flip-flop 630, the LOAD input of shift register 620 and the
23 inverting input of AND gate 640, from the output terminal of
24 AND gate 640 to input terminal 652 of OR gate 650, and from
25 the output terminal of OR gate 645 to the first input
26 terminal of OR gate 650. Finally, a SERIAL OUT line is
27 connected to the output terminal of OR gate 650, which is
28 recognized by the schematic capture program as being
29 connected to an external target.

30 The 7-bit UART transmitter circuit design operates as
31 follows. Upon power up or a UART RESET high signal, START
32 component 660 generates a high output signal which is passed
33 on the flow bus through JOIN2 component 661 to STATE
34 component 662 and at the next clock cycle to STATE component
35 664. The resulting high signals applied to the ACTIVE output
36 signal lines of STATE component 662 and STATE component 664
37 generate a high signal on the first and second input

1 terminals, respectively, of OR gate 645, which in turn
2 during at least two clock cycles in which STATE1 and STATE2
3 are active applies a high signal from its output terminal to
4 the first input terminal of OR gate 650. Note that the
5 ACTIVE output signal line from STATE component 664 remains
6 high until a high TEST signal is applied to IF component 665
7 While the TEST signal is low, IF component 665 continues to
8 apply the high signal from STATE component 664 to the FALSE
9 flow bus, thus maintaining the system in STATE 2 on
10 subsequent clock cycles. The resulting high signal generated
11 from the output terminal of OR gate 650 results in a high
12 SERIAL OUT signal, which constitutes a sequence of initial
13 "stop bits" of the UART transmission. A seven-bit signal is
14 then loaded on the DATA IN signal bus from external logic
15 (such as a host computer) to the D_IN terminal of data
16 register 610, and a high signal from the LOAD signal line is
17 applied to the CLOCK input terminal of data register 610 and
18 the C input terminal of D flip-flop 630. In response, the
19 seven bits are transmitted from the Q_OUT terminal of data
20 register 610 to the PAR_IN terminal of shift register 620,
21 and the D flip-flop applies a high signal to the TEST input
22 terminal of IF component 665. In response to the high TEST
23 signal, IF component 665 applies the high signal from STATE
24 component 664 to STATE component 666. When STATE component
25 666 applies a high signal to its ACTIVE output line, the high
26 signal is applied to the Reset terminal of D flip-flop 630
27 (causing the Q output to go low), to the LOAD terminal of
28 shift register 620 (causing seven bit data to be transferred
29 from the PAR_IN port to internal memory of shift register
30 620, and the least significant of the seven bits to be
31 applied to the LS_OUT terminal). The high signal from STATE
32 component 666 is also applied to the inverting input terminal
33 641 of AND gate 640 causing the SERIAL_OUT signal to go low,
34 which constitutes the "start bit" of the UART transmission.
35 As control passes through STATE components 667 through 673,
36 each of the seven bits is applied to the non-inverting input
37 terminal of AND gate 640. Note that because the inverting

1 input signal applied from STATE component 666 is low, thereby
2 enabling AND gate 640, the seven bits are duplicated on the
3 SERIAL OUT signal line. After the seven bits have been
4 transmitted, control is passed from STATE component 673
5 through JOIN2 component 661 to STATE component 662, thereby
6 resetting the UART transmitter circuit for transmission of an
7 additional seven bits.

8 Fig. 10b illustrates a pure schematic diagram
9 implementing the UART transmitter circuit of Fig. 10a. Note
10 that the combination of state flow components and schematic
11 components in Fig. 10a is easier to understand than the
12 purely schematic components of Fig. 10b.

13 The example of Fig. 10a is provided to illustrate the
14 use of four state flow components START, STATE, IF and JOIN.
15 Several modifications to the UART transmitter would make
16 more efficient use of the resources of a target FPGA. As
17 described below, the component PAUSE is introduced to
18 reduce the number of flip-flops required when using only the
19 four components shown in Fig. 10a.

21 PAUSE Component

22 Figs. 11a and 11b illustrate a PAUSE component which
23 generates a multicycle state condition. That is, the PAUSE
24 component generates a high ACTIVE signal for more than one
25 clock cycle.

26 As shown in Fig. 11b, the PAUSE component is implemented
27 using a D flip flop 710, a counter 720, a first AND gate 730,
28 a second AND gate 740, and an OR gate 750. AND gate 730
29 includes an inverted input 731 and a non-inverted input 732.
30 The ELEM=0 signal line from the PREV flow bus is connected to
31 the CLOCK input terminal of counter 720, to the C input
32 terminal of D flip-flop 710 and to the ELEM=0 signal line of
33 the NEXT flow bus. The ELEM=1 signal line from the PREV flow
34 bus is connected to the CLK_EN input of counter 720, the CE
35 input terminal of D flip-flop 710 and to the adjoining ELEM=1
36 signal line of the NEXT flow bus. The ELEM=2 signal line of
37 the PREV flow bus is connected to the ASYNC_CTRL input

1 terminal of counter 720, the R input terminal of D flip-flop
2 710 and to the ELEM=2 signal line of the NEXT flow bus. The
3 ELEM=3 signal line of the PREV flow bus is connected to the
4 LOAD input terminal of counter 720 and input terminal 751 of
5 OR gate 750. The FORCE input bus applies a multiple line
6 signal to the parallel D_IN terminal of counter 720. The
7 FORCE symbol is defined in a manner similar to the BUS DEF
8 symbol, and is explained further in the X-BLOX™ Design Tool
9 User Guide, mentioned above. The @CYCLES input value
10 assigned to the FORCE symbol indicates a binary number
11 applied over the FORCE bus to counter 720. The TERM_CNT
12 output terminal of counter 740 is connected to the inverting
13 input terminal 731 of AND gate 730 and to input 741 of AND
14 gate 740. The output of AND gate 730 is applied to input 752
15 of OR gate 750. The output of OR gate 750 is applied to the
16 D input of D flip-flop 710. The Q output of D flip-flop 710
17 is applied to the ACTIVE output line, to non-inverting input
18 terminal 732 of AND gate 730 and to an input terminal of AND
19 gate 740. The output terminal of AND gate 740 is applied to
20 the NEXT flow bus ELEM=3 line.

21 Operation of the PAUSE component begins when the ELEM=3
22 signal line of bus PREV goes high, (i.e., meaning when a
23 previous STATE element is activated). The high ELEM=3 signal
24 is applied to the LOAD input of counter 720, and the first
25 input of OR gate 750, resulting in a high signal applied to
26 the D input terminal of D flip-flop 710 for a duration of one
27 clock cycle. At the next rising clock transition, D
28 flip-flop 710 becomes SET, resulting in a high Q output
29 signal, and counter 720 is loaded with the value defined by
30 the FORCE @CYCLES variable. Assuming the @CYCLES is not
31 zero, a low signal is applied to the TERM_CNT output terminal
32 of counter 720 for the number of clock cycles specified by
33 the @CYCLES value (Counter 720 counts down from the specified
34 value-until counter 720 reaches zero). The low TERM_CNT
35 signal is applied to AND gate 740, which in turn applies a
36 low signal on the ELEM=3 signal line of the NEXT flow bus.
37 The low TERM_CNT signal is also applied to the inverting

1 input terminal 731 of AND gate 730. The high Q output of D
2 flip-flop 710 is applied to the ACTIVE signal line and to the
3 non-inverting input terminal 732 of AND gate 730, thereby
4 causing a high input signal to be applied to the D input
5 terminal of D flip-flop 710 through OR gate 750. At the end
6 of the clock cycle, the input signal on the ELEM=3 signal
7 line of the PREV flow bus becomes low, thereby applying a low
8 signal to the LOAD input terminal of counter 720. This low
9 signal causes counter 720 to operate in a "down count" mode
10 during subsequent clock cycles; that is, the counter
11 decrements from the @CYCLES value by one each clock cycle
12 until the counter reaches zero. In addition, although the
13 ELEM=3 signal line is low, a high signal continues to be
14 applied to the D input of D flip-flop 710 from OR gate 750
15 because of the high Q output signal, thereby causing the Q
16 output to continue to emit a high signal. Note that while
17 the Q output signal is high and the TERM_CNT signal is low,
18 AND gate 740 applies a low signal to the ELEM=3 signal line
19 of the NEXT flow bus. Finally, when counter 720 reaches
20 zero, the TERM_CNT output applies a high signal to AND gate
21 740 and to the inverting input of AND gate 730, thereby
22 causing AND gate 730 to apply a low signal to OR gate 750,
23 which in turn applies a low signal to the D input of D flip-
24 flop 750, which in turn causes the Q output of D flip-flop
25 750 to go low. However, before the Q output signal goes low,
26 the high signal applied from the Q output and from TERM_CNT
27 of counter 720 causes AND gate 740 to apply a high signal to
28 the ELEM=3 signal line of the NEXT flow bus.

29 As shown in Fig. 11c, the PAUSE component can be
30 incorporated into the flow diagram of Fig. 10a to replace
31 STATE components 666 to 673, thereby providing the desired
32 delay function in a more efficient manner.

34 FOR Component

35 Figs. 12a and 12b illustrate a FOR component which
36 creates iterative "FOR loops". The FOR component includes
37 two input flow paths, PREV and BACK, and two output flow

1 paths, LOOP and NEXT. In practice, the LOOP flow path is
2 directly or indirectly connected to the BACK flow path.
3 As indicated in Fig. 12b, the FOR component includes a
4 counter 810, AND gates 820 and 830, and OR gates 840 and 850.
5 The ELEM=0 signal line of the PREV flow bus is connected to
6 the CLOCK input terminal of counter 810, and to the ELEM=0
7 signal lines of the NEXT and LOOP flow buses. The ELEM=1
8 signal line of the PREV flow bus is connected to the ELEM=1
9 signal lines of the NEXT and LOOP flow buses. The ELEM=2
10 signal line of the PREV flow bus is connected to the
11 ASYNC_CTRL terminal of counter 810, and to the ELEM=2 signal
12 lines of the NEXT and LOOP flow buses. The ELEM=3 signal
13 line of the PREV flow bus is connected to the LOAD input
14 terminal of counter 810, to input 841 of OR gate 840, and to
15 input 851 of OR gate 850. In addition, the ELEM=3 signal
16 line of the BACK flow bus is connected to input 822 of AND
17 gate 820, to input 842 of OR gate 840, and to non-inverting
18 input 832 of AND gate 830. The LOOP bus uses the ELEM=0,
19 ELEM=1, AND ELEM=2 lines to control any intermediate
20 components in the loop, such as other STATE components.
21 However only the ELEM=3 line is provided by the BACK bus to
22 complete the loop.

23 The FOR component also includes a FORCE bus applying an
24 @CYCLES input value to the parallel D_IN input terminal of
25 counter 810. The @CYCLES variable is set by a user when the
26 FOR symbol is entered into a schematic, and the number is
27 entered on the FOR symbol. The TERM_CNT output terminal of
28 counter 810 is connected to input 821 of AND gate 820 and to
29 the inverting input 831 of AND gate 830. The output of AND
30 gate 820 is connected to the ELEM=3 signal line of the NEXT
31 flow bus. The output of AND gate 830 is connected to input
32 851 of OR gate 850. The output of OR gate 840 is connected
33 to the CLK_EN terminal of counter 810. Finally, the output
34 of OR gate 850 is connected to the ELEM=3 signal line of the
35 LOOP flow bus.

36 The FOR component operates as follows. When the ELEM=3
37 signal line of the PREV flow bus is high, the high signal is

1 applied to the LOAD terminal of counter 810 and to input 841
2 of OR gate 840. The high signal applied to the LOAD terminal
3 of counter 810 causes counter 810 to load the count value
4 defined by the @CYCLES variable. The high signal applied to
5 input 841 of OR gate 840 causes a high signal to be applied
6 to the CLK_EN terminal of counter 810. In addition, the
7 high signal on the ELEM=3 signal line of the PREV flow bus is
8 applied to input 851 of OR gate 850, which in turn applies a
9 high signal on the ELEM=3 signal line of the LOOP flow bus.
10 In typical applications of the FOR component, the LOOP flow
11 bus is connected to a state machine segment, or "loop", which
12 is connected to the BACK flow bus. The loop to which the
13 BACK flow bus is connected typically includes one or more
14 state flow components. Eventually, the state machine segment
15 applies a high signal onto the ELEM=3 signal line of the BACK
16 flow bus, which is then applied to input 822 of AND gate 820,
17 non-inverting input 832 of AND gate 830, and input 842 of OR
18 gate 840. If the CLK_EN signal on the ELEM=1 line of the
19 PREV flow bus is high, AND gate 843 responds to the high
20 output signal from OR gate 840, causing a high signal to be
21 applied to the CLK_EN input terminal of counter 810. (If the
22 CLK_EN signal is low, all logic elements including counter
23 810 are disabled.) Since the ELEM=3 signal on the PREV flow
24 bus is low (causing the LOAD input signal to be low), counter
25 810 decrements the count value at each clock cycle (note,
26 the clock signal is on the ELEM=0 line of the PREV flow bus)
27 as long as the CLK_EN input terminal of counter 810 is set
28 high by the ELEM=3 signal line of the BACK flow bus. While
29 the count value is greater than zero, the TERM_CNT output
30 signal is low, thereby preventing AND gate 820 from applying
31 a high signal to the ELEM=3 signal line of the NEXT flow bus.
32 Note that inverting input 831 of AND gate 830, in combination
33 with the high signal applied from the ELEM=3 signal line from
34 the BACK flow bus on input 852, causes a high signal to be
35 applied to the ELEM=3 signal line of the LOOP flow bus. When
36 the count value equals zero, the TERM_CNT signal goes high,
37 thereby causing AND gate 830 to apply a low signal on the

1 ELEM=3 signal line of the LOOP flow bus, and causing AND gate
2 820 to apply a high signal on the ELEM=3 signal line of the
3 NEXT flow bus. This high signal is applied for one clock
4 cycle, and moves the state machine out of the FOR loop.

6 WHILE Component

7 The WHILE component illustrated in Figs. 12c and 12d is
8 similar to the FOR component of Figs 12a and 12b. However,
9 instead of entering a number of cycles over which the FOR
10 component will loop, the WHILE component responds to a TEST
11 signal which is typically provided from another part of the
12 system. The WHILE component eliminates the need for counter
13 810 of Fig. 12b. The TEST signal which is applied to the
14 inverting input of AND gate 830 and the noninverting input of
15 AND gate 820 selects whether to activate the LOOP bus or the
16 NEXT bus.

18 CALL Component

19 In other embodiments of the present invention, flow
20 diagram subroutines are implemented by adding a fifth signal
21 line, ELEM=4, which transmits signals "upstream" (that is, in
22 a direction opposite to the signal directions of the ELEM=0
23 through ELEM=3 signal lines) along a flow bus. The fifth
24 signal line can be used to transmit a "return" signal which
25 indicates that a "called" subroutine has completed its
26 intended function. Note that if any symbols which must
27 return upstream data are used, then the STATE, IF, JOIN2,
28 JOIN3, PAUSE, FOR, and WHILE symbols which have been
29 described above must be modified to add the upstream data
30 line ELEM=4.

31 The CALL symbol is used for calling a subroutine and the
32 RETURN symbol is used for returning from the subroutine.
33 Figs. 13a and 13b show a symbol and equivalent circuit for a
34 CALL component. The CALL component circuit is similar to the
35 IF and FOR component circuits. However, an extra line ELEM=4
36 is added to provide a return signal, and flow of information
37 on the ELEM=4 line is in the opposite direction from that on

1 the ELEM=0 to ELEM=3 lines.

2 The CALL component operates as follows. As with other
3 symbols, a clock signal is provided on the ELEM=0 line of the
4 PREV flow bus and is applied to the ELEM=0 lines of the NEXT
5 and SUB flow buses. This clock signal is also applied to the
6 clock input C of flip flop 901. The ELEM=1 line of the PREV
7 flow bus carries a clock enable signal CLK_EN, which is
8 applied to the ELEM=1 lines of the SUB and NEXT flow buses
9 and also to the clock enable input CE of flip flop 901. As
10 with other symbols, the ELEM=2 line of the PREV flow bus
11 carries a reset signal, which is applied to the ELEM=2 lines
12 of the NEXT and SUB buses and also to the RD reset terminal
13 of flip flop 901. As with symbols described earlier, the
14 ELEM=3 line carries the ACTIVE signal. This signal is
15 applied directly to the ELEM=3 line of the SUB bus, so that a
16 high signal can be used to call a subroutine through the SUB
17 bus.

18 When a high ACTIVE signal appears on the ELEM=3 line of
19 the PREV flow bus, OR gate 902 applies a high signal to AND
20 gate 903. When the SUB bus is not providing a RETURN signal
21 (the usual state), the ELEM=4 line of the SUB bus is low.
22 Thus, AND gate 903 forwards the high signal from OR gate 902
23 to the D input of flip flop 901. At the next clock signal,
24 the high signal is applied by the Q output of flip flop 901
25 to AND gate 904, and enables AND gate 904 to activate the
26 ELEM=3 line of the NEXT bus in response to a high RETURN
27 signal from the SUB bus. Meanwhile the high Q output of flip
28 flop 901 is fed back on line 905 to OR gate 902. Thus AND
29 gate 903 will maintain a high signal to the D input of flip
30 flop 901 and thus a high Q output as long as the RETURN
31 signal on the ELEM=4 bus of the SUB bus remains low. Thus
32 gates 901, 902, and 903 are in a wait state ready to respond
33 to a high RETURN signal on the ELEM=4 line of the SUB bus.

34 When the subroutine is complete, this RETURN signal goes
35 high. The high RETURN signal applied to AND gate 904
36 produces a high signal on the ELEM=3 line of the NEXT bus.
37 Meanwhile, AND gate 903 applies a logical 0 to the D input of

1 flip flop 901, which on the next clock cycle appears at the Q
2 output, thus taking the CALL component out of its wait state,
3 and completing the CALL function.

5 RETURN Component

6 Fig. 14a shows the RETURN symbol and Fig. 14b shows its
7 equivalent circuit in which the ELEM=3 and ELEM=4 lines of a
8 PREV flow bus are connected together.

10 Example UART Transmitter Using FOR, CALL, PAUSE, AND RETURN 11 Components

12 Fig. 15 shows a UART transmitter in which the control
13 logic is represented by state machine components. The UART
14 transmitter has the ability to add START and STOP bits to a
15 data stream, and can be controlled to transmit either a
16 single byte in response to a LOAD signal or to transmit
17 multiple (in this example, eight) bytes in response to a LOAD
18 signal. One subroutine is used, therefore one RETURN symbol
19 is used. However, the subroutine is called from two different
20 points in the flow diagram, therefore two CALL symbols are
21 used.

22 Upon power-up or reset, a high UART reset signal
23 activates START component 560. This high UART reset signal
24 is also applied to the asynchronous ASY_CTRL input of shift
25 register 583, thereby resetting shift register 583. In
26 response to a high LOAD signal, data register 581 loads a
27 seven bit value from the DATA_IN line connected to its D_IN
28 input port. Thus, at the next clock cycle this value appears
29 at the Q_OUT terminal of data register 581. Also, in
30 response to the high LOAD signal at its clock terminal, FDRD
31 flip-flop 582 places the high VCC signal at its D input port
32 onto its Q output terminal. This high signal is applied to
33 IF component 565, causing the IF component to direct an input
34 signal to its TRUE output bus. When the UART ENABLE signal
35 goes high, shift register 583 is enabled to respond to
36 further clock signals. Thus, a UART clock signal following
37 the UART reset and UART enable signals causes shift register

1 583 to load data from the Q output of data register 581 into
2 its parallel input port. Shift register 583 will be used in
3 its serial output mode providing output signals from its
4 least-significant-value output port LS_OUT. Providing the
5 seven data bits on its output port will take seven clock
6 cycles of the UART clock signal. The state machine portion
7 of Fig. 15 counts these seven clock signals and also inserts
8 start and stop bits at the beginning and end of these seven
9 data bits.

10 The state machine may be set to transmit only a single
11 byte of data or to transmit multiple bytes of data. Consider
12 first the case in which a single byte of data will be
13 transmitted. On the first UART clock cycle after the UART
14 reset, JOIN3 component 561 activates STATE1 component 562.
15 On the second UART clock cycle after the reset signal, JOIN2
16 component 563 activates STATE2 component 564. If the LOAD
17 signal has not yet been applied, IF component 565 will have a
18 low value on its TEST input and its FALSE bus will apply the
19 ACTIVE signal from STATE2 component 564 to JOIN2 component
20 563, thus returning the state machine to the state
21 represented by STATE2 component 564. This cycling continues
22 until a high LOAD signal causes FDRD flip-flop 582 to apply a
23 high Q output signal to the TEST input of IF component 565.
24 When the LOAD signal is applied, data on the DATA_IN line are
25 loaded into data register 581. Assuming that the SINGLE XMIT
26 line carries a high signal, the signal to the TEST input of
27 IF component 566 is high, and a TRUE bus ACTIVE output signal
28 will be received by call component 567, which calls a
29 subroutine through JOIN2 component 571, moving the state
30 machine to STATE component 572. While the machine is in
31 STATE 572, a start bit is added to the data stream by
32 applying a high active signal to the inverted input terminal
33 of AND gate 584, causing AND gate 584 to apply a low output
34 signal to OR gate 585.

35 Since OR gate 586 is not receiving a high input from any
36 of the active signals of states 562, 564, or 570, OR gate 586
37 applies a low output signal to the other input terminal of OR

1 gate 585. Thus, the SERIAL_OUT line carries a low output
2 signal which serves as a start bit for the data to follow.
3 In the next clock cycle, the state machine moves from STATE
4 component 572 to PAUSE component 573, where the state remains
5 for the next seven cycles while seven data bits are being
6 transmitted. Meanwhile, the data shifted in parallel into
7 shift register 583 are shifted out in series through AND gate
8 584 and OR gate 585. Since STATE component 572 is not
9 currently active, the low output signal from STATE component
10 572 applied to the inverting input terminal of AND gate 584
11 causes AND gate 584 to forward the signal received from shift
12 register 583 to the upper terminal of OR gate 585. Since the
13 lower terminal of OR gate 585 is receiving a low input signal
14 from OR gate 586, the data on its upper input terminal are
15 forwarded to the SERIAL_OUT terminal and transmitted. After
16 seven data bits have been transmitted, PAUSE component 573
17 moves the state machine to its next stage, which is RETURN
18 component 574. As stated above, RETURN component 574 simply
19 forwards the high signal received from PAUSE component 573
20 back through the return path to CALL component 567. As
21 discussed above in connection with Figs. 13a and 13b, when
22 the CALL component calls a subroutine, the Q output of flip
23 flop 901 places a high signal on one input terminal of AND
24 gate 904. In this manner, AND gate 904 waits for a high
25 RETURN signal to activate the ELEM=3 line of its NEXT bus.
26 Thus, a high RETURN signal on the ELEM=4 line of the SUB bus
27 produces a high ELEM=3 signal on the NEXT flow bus, thus
28 moving the state machine to the next state. This high RETURN
29 signal is also applied to the inverting input terminal of AND
30 gate 903 and produces a low signal to the D input terminal of
31 flip flop 901. Thus, on the next clock cycle, a low Q output
32 signal appears at the upper input of AND gate 904, which
33 causes the ELEM=3 line of the next bus to go low again and
34 which places the CALL component into a state of not waiting
35 for a RETURN signal. The low Q output of flip flop 901 in
36 combination with a low ELEM=3 signal from the PREV flow bus
37 causes OR gate 902 to apply a low signal to AND gate 903.

1 Thus, the Q output of FDRD flip-flop 901 is low, and the call
2 component is no longer waiting for a return signal from a
3 subroutine.

4 Returning now to Fig. 15, CALL component 567 forwards a
5 high signal on its NEXT flow bus through JOIN3 component 561
6 to STATE1 component 562. This results in a high ACTIVE
7 output signal on the ELEM=3 line of STATE component 562,
8 which in turn results in a high signal being applied by OR
9 gate 586 to OR gate 585, which in turn results in a high STOP
10 bit applied to the SERIAL_OUT line by OR gate 585. On the
11 next clock cycle, the state machine moves through JOIN2
12 component 563 to STATE component 564, which again applies a
13 high output signal to OR gate 586, which is in turn forwarded
14 to OR gate 585 to generate a second STOP bit. The machine
15 remains at STATE2 component 564 with further high STOP bits
16 until another LOAD signal is applied to the clock inputs of
17 data register 581 and FDRD flip-flop 582.

18 To transmit eight bits in a single burst, the SINGLE
19 XMIT line is held low. Thus, when a high LOAD signal causes
20 a high TEST signal to be applied to IF component 565 and the
21 state moves to IF component 566, IF component 566 forwards an
22 ACTIVE signal on its FALSE bus to FOR component 568
23 (discussed above in connection with Figs. 12a and 12b). In
24 the example of Fig. 15, the user has set the FOR component
25 568 to repeat for 8 cycles. Thus for 8 consecutive times,
26 the system will move from FOR component 568 to CALL component
27 569, which calls the subroutine through JOIN2 component 571,
28 which at STATE component 572 adds a START bit (low) to the
29 data stream, then moves to PAUSE component 573, which pauses
30 for seven cycles while the seven data bits are being
31 transmitted, and RETURNS. The RETURN signal is transmitted
32 back through JOIN2 component 571 to both CALL components 567
33 and 569. But only CALL component 569, which called the
34 subroutine, is waiting for the RETURN signal (as discussed
35 above). When CALL component 569 receives the RETURN signal,
36 it activates STATE component 570 (i.e., causes STATE
37 component 570 to send a high signal from its ACTIVE

1 terminal), which causes OR gate 586 to send a high STOP bit
2 to OR gate 585 which is forwarded to the SERIAL_OUT port.
3 After 8 subroutine calls, FOR component 568 activates its
4 NEXT bus, which applies a high ACTIVE signal to JOIN3
5 component 561 and moves the system to STATE1 component 562.
6 STATE1 component 562 causes OR gate 586 to send another high
7 STOP bit. The system moves to STATE2 component 564, where it
8 remains due to the loop from IF component 565 until another
9 LOAD signal is received.

11 STOP Component

12 A STOP component, shown in Figs. 16a and 16b provides
13 simply a ground connection to the ELEM=4 line. In this
14 manner, any component connected to a STOP component carries a
15 constant low signal on its RETURN line. Thus a RETURN line
16 connected to a STOP component never activates another state
17 in the state machine.

19 Multiple States Active

20 The present invention also provides additional
21 capability in a complex circuit design. For example, in one
22 embodiment a specialized state machine having several
23 independent logic paths is implemented using coding similar
24 to one-hot encoding. As with one-hot encoding, a high flip
25 flop output from a state machine component indicates the
26 state is active. However, more than one flip flop output may
27 be high simultaneously, indicating that more than one state
28 is active. To implement several parallel paths in accordance
29 with the present invention simply requires connecting a flow
30 bus from one state flow symbol output to two or more state
31 flow symbol inputs. The result is a branching of the state
32 sequence such that each branch has an active state. In such
33 a case, each branch typically controls an independent part of
34 the circuit design. For example, a state machine which
35 accesses external data can have an independent branch for a
36 cache controller and another branch for the primary
37 algorithm. There may be a need for and a provision for

1 synchronization between the two or more branches. Several
2 methods will be described for activating multiple states of a
3 state machine.

5 SPAWN Component

6 The SPAWN component allows for branching to two
7 simultaneously active states. Fig. 17a shows a SPAWN symbol
8 and Fig. 17b shows the equivalent circuit of a SPAWN
9 component. Like the CALL component, the SPAWN component uses
10 an ELEM=4 bus line for carrying up-stream information. OR
11 gate 171 returns the logical OR of the signals on the ELEM=4
12 buses of the LEFT and RIGHT buses. This SPAWN component
13 facilitates implementation of branching state machines, in
14 which separate branches can operate in parallel.

15 Branching state machines provide a powerful mechanism
16 for controlling separate but linked parts of a system. For
17 example, a coprocessor with on-chip cache can benefit from
18 the use of a branching state machine by spawning off a new
19 branch whenever a memory fetch takes place. One spawned
20 branch can implement a cache algorithm, while another branch
21 continues processing the instruction stream. Active outputs
22 of the cache branch are input to the processing branch at
23 places where the processing branch must synchronize with the
24 cache branch. Upon terminating its task, the cache branch
25 "falls off the end" of the state diagram. That is, the last
26 state of the cache branch has nothing, or a STOP symbol,
27 connected to its NEXT flow bus.

28 When a SPAWN symbol is used, the system no longer has
29 the characteristic of having only one state active at one
30 time. Care must be taken to insure that the separate
31 branches are truly independent and do not corrupt each
32 other's integrity. In particular, a common subroutine should
33 not be called from multiple branches. Also the designer must
34 avoid feeding a JOIN symbol from different branches unless
35 the system is designed to accommodate this. Otherwise, more
36 than one state may be active within a single branch, and this
37 may produce an unpredictable or unanticipated result if the

1 designer has not planned for multiple active states in the
2 design.

4 UART Example Using SPAWN

5 Fig. 18 shows an example using the SPAWN component along
6 with START, STATE, IF, PAUSE, and STOP state flow symbols
7 discussed earlier. In this example, a UART receiver spawns
8 off a new branch to transmit the received data. This allows
9 the receiver to work in parallel with the transmitter as
10 would be required in continuous data flow. Upon power-up or
11 reset, START component 860 is asserted. At the next cycle of
12 the UART clock, the system shifts through JOIN3 component 861
13 to STATE component 862. Shift register 881 receives data
14 serially on its MS_IN port, and sends data in parallel on its
15 PAR_OUT port.

16 As is well known, UART receivers are synchronized to
17 data being received from a remote terminal as well as to a
18 local clock signal. Thus the logic for controlling a UART
19 receiver responds to received data as well as to a local
20 clock signal. As long as the data signal on the SERIAL_IN
21 port is a logical 1, no start bit will be recognized, and IF
22 component 865 activates its FALSE bus so that the state
23 machine remains at STATE component 862.

24 However, if the SERIAL_IN signal goes to a logical 0,
25 IF component 865 activates the TRUE bus, moving the state
26 machine to PAUSE component 866. In addition to pausing for 7
27 cycles of the UART clock, PAUSE component on its ACTIVE
28 output line enables shift register 881 to receive the UART
29 clock signal on its CLOCK terminal. This CLK_EN signal
30 enables the next 7 clock signals so that the next 7 values on
31 the SERIAL_IN terminal are read into the MS_IN port of shift
32 register 881.

33 After these seven bits have been read into shift
34 register 881, PAUSE component 866 disables shift register 881
35 so that serial bits do not continue to be read in. PAUSE
36 component 866 also provides a high signal to SPAWN component
37 867 SPAWN component 867 provides two output signals. On its

1 LEFT flow bus, SPAWN component provides this high signal to
2 JOIN3 component 861, activating STATE1 component 862. On its
3 RIGHT flow bus, SPAWN component 867 activates STATE2
4 component 868.

5 STATE2 component 868 is the first of several components
6 which control the transmitting function of the UART. In the
7 next two UART CLOCK cycles, the transmit branch of the system
8 moves through STATE3 component 869 to STATE4 component 871.
9 The ACTIVE (high) output of STATE4 component 871 causes shift
10 register 882 to load data which are present on its PAR_IN
11 input port. This ACTIVE signal also produces a low output
12 from AND gate 883, and causes NOR gate 885 to provide a low
13 signal to OR gate 884. This results in a low SERIAL_OUT
14 signal from OR gate 884, which serves as a START bit for the
15 transmitted signal. At the next clock cycle, the transmit
16 branch moves to PAUSE component 872, which provides a high
17 ACTIVE signal to NOR gate 885 for the next seven cycles. The
18 high signal to NOR gate 885 produces a low input to OR gate
19 884 so that data received on the upper input terminal of OR
20 gate 884 from AND gate 883 are passed to the SERIAL_OUT line
21 for transmission by OR gate 884. When the 7 cycles have been
22 completed, the transmit branch of the state machine is
23 terminated by STOP component 873.

24 As the transmit branch of the state machine moves to
25 STATE2 component 868, the receive branch of the state machine
26 moves to STATE1 component 862. IF component 865 maintains
27 the receive branch at STATE1 component 862 waiting for the
28 value on the SERIAL_IN line to go from logical 1 to logical 0
29 in order to again move to PAUSE component 866 for receiving
30 another 7 bits of data. If data are being transmitted
31 steadily on the SERIAL_IN port, both receive and transmit
32 branches of the state machine will be active simultaneously,
33 with a byte of data moving from shift register 881 to shift
34 register 882 as controlled by the START and STOP bits of the
35 SERIAL_IN data stream and the UART clock through PAUSE
36 component 866. Since the RIGHT branch of the SPAWN symbol
37 terminates in a STOP symbol, no discrepancies result from

1 having two simultaneously active states.

3 Pipeline Controller Having Several Active States

4 Another example of a state machine having more than one
5 active state is a pipeline controller. Pipelined logic
6 comprises several stages of logic in which one stage is
7 separated from the next by a flip flop and the flip flops are
8 commonly clocked so that an output signal from a first stage
9 enters a second stage at the same time an output signal from
10 the second stage enters a third stage. That way, a long,
11 time-consuming, string of logic operations may be divided
12 into pieces and the pieces operated simultaneously like
13 stages in an assembly line, with samples being processed at a
14 throughput rate determined by the time interval required by
15 the slowest piece, not the time interval required to compute
16 the entire string.

17 Such a pipelined system may be controlled by a
18 "pipelined" state machine. In such a state machine there is
19 one active state controlling each stage of the pipeline
20 design. Thus with pipelining there are several
21 simultaneously active states on one data path through one
22 state machine. Pipelined state machines can be implemented
23 by providing flow bus connections which allow control signals
24 to be entered at selected points on the flow bus. To
25 activate more than one state, an arrangement similar to the
26 JOIN symbols of Figs. 8a-8c can be used. Instead of the
27 RIGHT bus of Fig. 8b, a single input line is provided, and
28 connected to OR gate 410. Then when either this single input
29 line carries a high signal or when line ELEM=3 of the PREV
30 bus carries a high signal, the ELEM=3 line of the NEXT bus
31 goes high. Thus the single input line may be used to
32 activate a second state.

34 ACTIVATE and TAP Symbols

35 The ACTIVATE component induces a state to become active
36 by an event external to the state diagram. This symbol may
37 be used to cause a state machine to have more than one active

1 state at one time. Figs. 19a and 19b show an ACTIVATE symbol
2 and its corresponding circuit. A signal from a single line,
3 the ACTIVATE line, is combined by OR gate 191 with the ELEM=3
4 line of the PREV bus. Thus, if either is high, a high signal
5 is applied to the ELEM=3 line of the NEXT flow bus. In
6 another embodiment, the ACTIVATE circuit includes an ELEM=4
7 line which carries a RETURN signal, as discussed above.

8 The TAP symbol and its corresponding circuit are shown
9 in Figs. 19c and 19d. The TAP symbol complements the
10 ACTIVATE symbol by providing a TAP output signal from the
11 ELEM=3 line. The symbol is useful for synchronizing
12 functions, for example generating a synchronous load signal
13 for an external register. If the external register needs to
14 be loaded at the same time a specific state becomes active,
15 without regard to any previous state, then that state is
16 preceded in the state flow diagram with a TAP symbol, and the
17 TAP symbol's ACTIVATE output signal is used as the register
18 load command input signal.

20 Example Design Using Reentrant State Machine

21 When using a reentrant state machine (a machine with
22 multiple states active, and with state activation from more
23 than one source), it is important to control the relationship
24 between active states so that no logical conflict occurs.
25 The following example illustrates the problem and its
26 solution.

27 The example is a design of a system for real time
28 compression of a digital video stream from 10 bits to 8 bits.
29 The design requirements are such that a simple linear
30 conversion (i.e. simply dividing the numbers by 4) is
31 unacceptable. Most of the interesting information in the
32 signal lies near the middle of the range of the incoming
33 numbers, around the value 512. Also, it is unacceptable to
34 simply throw away values near the limits of the range, near 0
35 and 1024, since this would result in noticeable image
36 distortion. The desired solution is to use a non-linear
37 transfer function where values near the center are

1 uncompressed, but values above and below the center by some
2 amount are compressed.

3 Fig. 20 shows an acceptable transfer function. The x
4 axis represents values before transfer and the y axis
5 represents values after transfer. Values between $x=0$ and
6 $x=439$ will be divided by 8. That is, the formula $y = x/8$
7 will be used. Values between $x=440$ and $x=585$ will have 384
8 subtracted, that is the formula $y = x - 384$ will be used.
9 And values between $x=586$ and $x=1024$ will be converted by the
10 formula $y = 128 + x/8$.

11 The design uses pipelining to achieve high throughput,
12 and uses a state machine with multiple active states,
13 maintaining the data between stages of the pipeline in lock
14 step.

15 Fig. 21 shows a state machine and other hardware used in
16 the design of the data compression device. On the left side
17 of the figure are components of the state machine portion of
18 the design, and along the right side are logic elements which
19 are controlled by the state machine. Not shown in the
20 diagram is a global clock line which provides a CLOCK signal
21 to data registers 2130, 2133, 2138, and 2140. The same CLOCK
22 signal which is propagated from the CLOCK input of START
23 component 2160 through the various ACTIVATE, STATE, IF, and
24 JOIN components of the state machine portion of the design
25 also feeds the data register CLOCK inputs. The line is
26 omitted in order to simplify the drawing, but is, of course,
27 important for proper operation of the circuit.

28 When a new sample of data is to be compressed, a high
29 NEW SAMPLE signal is applied to ACTIVATE component 2161 at
30 the same time the sample data are applied to the DATA_IN line
31 of data register 2130. As discussed in connection with Fig.
32 19b, the ACTIVATE component forwards this high signal
33 asynchronously to its NEXT bus. Thus on the first clock
34 cycle, a high ELEM=3 signal appears on the PREV bus of STATE
35 component 2162.

36 At the second clock cycle, STATE component 2162 applies
37 a high ELEM=3 signal to its NEXT bus as data register 2130 is

1 applying the received data to its Q_OUT terminal. On this
2 second clock cycle, comparators 2131 and 2132 test the data
3 to determine whether the data value is greater than or equal
4 to 439 and less than 585, respectively. Results of these
5 tests are applied to the TEST inputs of IF components 2163
6 and 2164, respectively (see Fig. 20). If the data value is
7 in the low range, a low value on the A_GE_B terminal of
8 comparator 2131 results in IF component 2163 activating its
9 FALSE bus which in turn activates STATE component 2165.
10 Otherwise, IF component 2163 activates IF component 2164. If
11 comparator 2132 indicates the data value at A is less than
12 585 (therefore in the middle range) the high signal on the
13 A_LT_B output terminal applied as a TEST signal to IF
14 component 2164 activates the TRUE bus, thus applying a high
15 ELEM=3 signal to the PREV bus of STATE component 2166. A low
16 signal on the A_LT_B output terminal of comparator 2132
17 causes IF component 2164 to activate its FALSE bus, which
18 applies a high ELEM=3 signal to STATE component 2167.
19 Meanwhile the data value is applied to the D_IN terminal of
20 data register 2133.

21 At the third clock cycle, one of the three STATE
22 components 2165, 2166, and 2167 applies a high value to the
23 ELEM=3 line of its NEXT bus. If STATE component 2166 was
24 activated (because the data value was in the middle range), a
25 high ACTIVE output from STATE component 2166 is also applied
26 to inverter 2151, resulting in a low control signal to
27 multiplexer 2136. This low signal causes multiplexer 2136 to
28 place the original data value on its output terminal, which
29 is applied to the D_IN terminal of data register 2138.
30 Otherwise, if STATE component 2166 was not activated,
31 inverter 2151 applies a high value to multiplexer 2136. In
32 this case, a new value calculated by element 2134, which is
33 the original data value divided by 8, is placed on the output
34 terminal of multiplexer 2136. (Recall that the data
35 compression algorithm being implemented divides by 8 the
36 values at the upper and lower parts of the range.) This
37 quotient is applied to the D_IN terminal of data register

1 2138.

2 At the fourth clock cycle, one of STATE components 2168,
3 2169 and 2170 is activated, again depending upon the range of
4 the data value to be compressed. If the data value is in the
5 low range, STATE component 2168 causes multiplexer 2137 to
6 place a zero value on its output bus. If the data value is
7 in the high or middle range, STATE component 2168 will not be
8 activated and will cause multiplexer to place onto its output
9 bus the value on its upper input bus. If the data value is in
10 the high range, STATE component 2170 causes multiplexer 2135
11 to place the value "128" on its output bus. This value is
12 then forwarded by multiplexer 2137 to the B input terminal of
13 ADD_SUB accumulator 2139. If the data value is in the middle
14 range, STATE component 2170 provides a low control value to
15 multiplexer 2135 so that multiplexer 2135 places the value
16 "384" on its output bus, this value being forwarded by
17 multiplexer 2137 to ADD_SUB accumulator 2139. Also if the
18 data value is in the middle range, STATE component 2169
19 causes ADD_SUB accumulator to operate in its subtract mode,
20 subtracting the value on its B input terminal (384) from the
21 value on its A input terminal (the original data value). The
22 difference is placed on the output terminal of ADD_SUB
23 accumulator 2139, and thus on the D_IN terminal of data
24 register 2140. The other possible results can be understood
25 from the above description.

26 At the fifth clock cycle, data register 2140 applies the
27 value on its D_IN bus to the DATA_OUT bus as the compressed
28 data signal.

29 While a first data value is progressing through the
30 system, subsequent values may also be simultaneously
31 progressing through the system. In this case, while a first
32 data value is being processed by the logic between data
33 register 2138 and data register 2140 under the control of
34 STATE components 2168, 2169, and 2170, a second data value is
35 being processed by the logic between data registers 2133 and
36 2138 as controlled by STATE components 2165, 2166, and 2167,
37 and a third data value is being processed by the logic

1 between data registers 2130 and 2133 under the control of
2 STATE component 2162 and IF components 2163 and 2164. A
3 fourth data value may be waiting at the DATA_IN input of data
4 register 2130 for the next clock cycle. Thus with
5 pipelining, even though the calculation requires five clock
6 cycles, a new data value may appear at the DATA_OUT terminal
7 every clock cycle. And because the three paths through the
8 state machine are synchronized with each other and with the
9 data registers of the pipeline, there is no conflict between
10 multiple active states.

11 In another implementation of the design, comparators
12 2131 and 2132 are eliminated by replacing IF components 2163
13 and 2164 with the comparator-IF components discussed below in
14 connection with Figs. 24a and 24b.

16 Invalid State Elimination

17 Invalid states occur under certain conditions including
18 but not limited to power glitches, improper power-up actions,
19 or race conditions on asynchronous inputs. The most likely
20 producer of invalid states in a state machine is activation
21 of both branches of a conditional branch component. Another
22 is failure to become inactive when the next state is
23 activated. A state machine implementation which fails to
24 account for invalid states may get into an infinite loop or
25 provide unpredictable results. The method and symbols of the
26 present invention offer the user several options of
27 eliminating invalid states if they should occur.

28 The state machine components described above use a high
29 flip flop output to represent an active state. In the
30 reentrant state machines such as shown in Fig. 21, several
31 states may legitimately be active simultaneously. The
32 invalid state elimination structure and method allow only one
33 active state. Therefore, the invalid state elimination
34 structure and method would not be used for such a design.
35 However, in many state machines, only one state should be
36 active, and the invalid state elimination method and
37 structure assure this result.

1 Two structures and methods for eliminating invalid
2 states will be described. The first form of protection
3 provides recovery from multiple states being active in a loop
4 and prevents descendants of an active state from becoming
5 active simultaneously with the active state. A second form
6 of protection prevents activation of both branches of a
7 conditional branch component. In one embodiment, both forms
8 of protection are used, though each may be used separately.

9 The invalid state elimination logic will be described in
10 connection with Figs. 22a through 23. To implement both
11 methods of invalid state elimination logic, three new signal
12 lines are added to the flow bus of components, and additional
13 logic is added to the RESET (ELEM=2) path of the STATE
14 component. An XR signal line prevents more than one branch of
15 a decision tree from becoming active. An R1 reset line
16 causes STATE flip flops beyond the next state to be reset.
17 An ER signal allows the Q output of the first STATE-IS
18 component past a JOIN2-IS component to force a reset on all
19 states beyond itself. This assures that all loops are
20 restricted to a single active state.

21 For comparison, Figs. 22a, 22c, 22e and 22g illustrate
22 in simplified form the circuits of Figs. 6b, 7b, 8b, and 9b
23 respectively. In their simplified forms, Figs. 22a, 22c,
24 22e and 22g do not show the clock and clock enable lines.
25 Also, the bus is not shown, rather the individual bus lines
26 are illustrated. Figs. 22b, 22d, 22f, and 22h illustrate, in
27 this same simplified form, components which can be included
28 in a library of components to provide recovery from invalid
29 active states. Like Figs. 22a, 22c, 22e and 22g, Figs. 22b,
30 22d, 22f, and 22h do not show the clock or clock enable
31 lines, which are of course present in the actual circuit.

32 Fig. 22d illustrates a STATE-IS component. Note that
33 the STATE-IS component of Fig. 22d uses different
34 nomenclature from the STATE component of Fig. 22c.
35 Specifically, the active state AS line corresponds to the E=3
36 line of Fig. 22c. Also, In Fig. 22d, two reset lines R0 and
37 R1 and logic for eliminating invalid active states replace

1 the E=2 line of Fig. 22c. OR gate 225 receives the two reset
2 signals R0 and R1 from its PREV bus and generates an R0 reset
3 signal on its NEXT bus. AND gate 226 receives the Q output
4 signal and a reset enabling signal ER from its PREV bus, and
5 generates the second reset signal R1 on its NEXT bus.
6 Finally, the Q output signal from flip flop 224 is fed back
7 as the XR signal to the PREV bus.

8 Fig. 22h illustrates an IF-IS component. The IF-IS
9 component can be connected to work with the STATE-IS
10 component of Fig. 22d. Like the IF component of Fig. 22g
11 which uses AND gates 228 and 229 to select which of the TRUE
12 and FALSE E=3 lines to activate, the IF-IS component of Fig.
13 22h uses AND gates 232 and 234 to select which AS output line
14 to activate. To prevent both outputs of an IF component from
15 becoming active, the IF-IS component of Fig. 22h applies an
16 XR signal (which will be applied by the Q output of a
17 STATE-IS symbol attached to the TRUE bus, see Fig. 22d) to OR
18 gate 233. OR gate 233 also receives the R0 reset signal from
19 a previous component and generates an R0 output signal on its
20 FALSE bus if either the R0 input signal is high or if the XR
21 signal is high.

22 Fig. 22b shows a START-IS component. This START-IS
23 component is to be used with the STATE-IS, IF-IS, and
24 JOIN2-IS components. The START-IS component receives a reset
25 signal R from another part of the system and in response
26 generates a high AS output signal Q from flip flop 223 to
27 start the state machine. This R signal is also applied to
28 the R0 output terminal, and thus resets all STATE-IS
29 components in the state machine.

30 Fig. 22f shows a JOIN2-IS component to be used with the
31 STATE-IS and IF-IS components. Like Fig. 22e, the JOIN2-IS
32 component of Fig. 22f uses OR gate 231 to provide a high AS
33 output signal if either of the AS input signals is high. The
34 JOIN2-IS component of Fig. 22f does not forward the RIGHT R0
35 reset signal. Nor does it forward the R1 reset signal from
36 either the RIGHT or PREV input buses.

37 Operation of this elimination logic is as follows.

1 Referring to Fig. 22d, a high signal on the R0 reset line
2 resets flip flop 224, thereby producing a low Q output
3 signal. The high R0 signal is also propagated by OR gate 225
4 to the NEXT bus so that one high R0 signal produces a low Q
5 output signal in any flip flops downstream. The Q output
6 signal, in addition to driving the active state AS line, is
7 also applied to AND gate 226. The other input to AND gate
8 226 comes from the ER line of the PREV input bus. If there
9 is a JOIN2-IS component driving the PREV bus, or one or more
10 IF-IS components preceded by a JOIN2-IS component, then the
11 ER line will be forced high, allowing the Q output signal of
12 the STATE-IS component next downstream to appear at the
13 output of its AND gate 226. If, however, the PREV bus is
14 driven from a STATE-IS component, then the ER line of the
15 PREV bus is low, forcing a low R1 output of AND gate 226.
16 Thus the Q output of flip flop 224 in one STATE-IS component
17 causes reset of all flip flops 224 in downstream STATE-IS
18 components beyond the immediate next STATE-IS component, and
19 further to the first JOIN2-IS component. Thus in any loop,
20 there is a state which, when active, forces all other states
21 in that loop to be inactive.

22 The Q output terminal of flip flop 224 further provides
23 the XR feedback reset signal. This XR feedback signal
24 eliminates invalid states resulting from an IF component.
25 Thus, when the IF-IS component is connected at its TRUE bus
26 to a STATE-IS component such as shown in Fig. 22d, a high Q
27 output signal which generates a high XR output signal causes
28 the IF-IS component to generate a high R0 output signal on
29 its FALSE bus. This high R0 signal causes a STATE-IS
30 component connected to the FALSE bus of the IF-IS component
31 to be reset to generate a low Q output signal, thereby
32 indicating an inactive state. Thus STATE-IS components
33 connected to both the TRUE and FALSE buses of the IF-IS
34 component will never both be active simultaneously. Note
35 that the IF-IS component also passes the XR signal applied to
36 its FALSE flow bus back to its PREV flow bus to facilitate
37 nesting of IF-IS components (explained in connection with

1 Fig. 23).

2 It is important that the R0 and R1 lines of the JOIN2-IS
3 component not propagate to the NEXT bus of the JOIN2-IS
4 component. In fact, this discontinuity prevents the invalid
5 state elimination logic from looping a reset signal back to
6 the state which should legitimately be active. Instead, the
7 JOIN2-IS component applies a high signal VCC to its output ER
8 line. This high ER line enables the next STATE-IS component
9 to force other STATE-IS components downstream to be reset.
10 Therefore, a loop in which two STATE-IS components and a
11 JOIN2-IS component are placed can recover from multiple
12 active states.

13

14 In another embodiment, a global control line (not shown)
15 is added to the flow buses, and when pulled low, disables the
16 invalid state elimination logic. Such a line is useful for a
17 system in which some multiply active states are desirable.
18 Indeed many other embodiments of these functions will become
19 obvious to those skilled in the art in light of the
20 description here.

22 Example Using Invalid State Elimination Logic

23 Fig. 23 shows an illustrative portion of a state diagram
24 in which invalid state elimination components are used.
25 JOIN2-IS component 2301 combines active state AS signals from
26 its RIGHT and PREV flow buses to activate STATE-IS component
27 2302. Nested IF-IS components 2303 and 2304 select one of
28 STATE-IS components 2305, 2306, and 2307 as determined by
29 TEST signals 2311 and 2312.

30 As shown in Fig. 23, JOIN2-IS component 2301 forwards
31 the R0 reset signal from its PREV bus to its NEXT bus but
32 does not forward reset signals on its R1 reset line. Thus a
33 RESET signal from a START-IS component (not shown) propagates
34 through JOIN2-IS component 2301. Note that a high R0 reset
35 signal will cause a reset of flip flop 224 of STATE-IS
36 component 2302 and will cause OR gate 225 of STATE-IS
37 component 2302 to send a high R0 reset signal to IF-IS

1 component 2303. This high R0 reset signal in turn applies
2 high R0 signals to both its TRUE and FALSE output buses.
3 (Locations of the TRUE and FALSE output buses are shown in
4 Fig. 22h.) This high R0 signal further propagates through
5 STATE-IS components 2305, 2306, 2307, and 2308. Thus all
6 STATE-IS flip flops are reset in response to a high R0 reset
7 signal.

8 Since JOIN-IS component 2301 applies a high ER signal to
9 AND gate 226 of STATE-IS component 2302, a high Q output
10 signal from flip flop 224 of STATE-IS component 2302 causes
11 AND gate 226 to apply a high R1 output signal through IF-IS
12 components 2303 and 2304 to OR gates 225 of STATE-IS
13 components 2305, 2306, and 2307. As can be seen with
14 STATE-IS component 2308, the JOIN-IS component causes a
15 RESET of flip flops in STATE-IS components (such as 2308)
16 which are downstream by two from a STATE-IS component (such
17 as 2302) which generates a high R1 output signal.

18 Looking next at IF-IS components 2303 and 2304, assume
19 that TEST line 2311 carries a low signal and TEST line 2312
20 carries a high signal, but not quite high enough. This means
21 a high Q output signal for STATE-IS component 2302 will
22 activate STATE-IS component 2307. In the event that the high
23 TEST signal on line 2312 did not properly produce a low input
24 to AND gate 232 of IF-IS component 2304, and the AS line
25 improperly applied a high D input to flip flop 224 of
26 STATE-IS component 2306, the high Q output from STATE-IS
27 component 2307 applied by the XR line to OR gate 233 of IF-IS
28 component 2304 would reset flip flop 224 of STATE-IS
29 component 2306 (as well as any other STATE-IS components
30 downstream, not shown), thus preventing an erroneous result
31 from an insufficiently high TEST signal on line 2312.

33 Additional Components

34 For additional flexibility, components having certain
35 additional features are included in a library. Instead of
36 some of the values being entered by a user, components can
37 take values from input buses, to allow a value to be

1 generated within the system and used as a test value.
2 Further, a test bus related to one or a few components can

4 be merged with a flow bus related to the entire system, as
5 will be discussed below in connection with the VECTOR
6 components.

8 Comparator IF Component

9 Figs. 24a and 24b show a symbol and circuit for a
10 comparator-if or CIF component. This component is similar to
11 the IF component shown in Figs. 9a and 9b. However, the CIF
12 of Figs. 24a and 24b branches to the TRUE bus only if a value
13 on a TEST bus matches a value entered on the CIF symbol. As
14 shown in Fig. 24b, COMPARE block 231 compares a value on the
15 A input bus to a value on the B input bus and can be
16 connected to provide several outputs. For example, on the
17 A_EQ_B output terminal, COMPARE block 231 provides a high
18 output when A equals B. On the A_NE_B output terminal a high
19 output is provided when A is not equal to B. On the A_LT_B
20 output terminal the signal is high when A is less than B.
21 Other inequalities cause high signals on the remaining
22 terminals. As shown, AND gates 510 and 520 are connected to
23 the A_EQ_B output terminal and thus receive a high signal
24 when A is equal to B.

25 Clearly many other IF-type components can also be
26 provided by including components having different
27 connections. For example a "greater than" comparator
28 component is implemented by connecting AND gates 510 and 520
29 to the A_GT_B output terminal of COMPARE block 231.

31 Coded CALL Component and SUBROUTINE Component

32 A more advanced method of handling subroutine calls is
33 needed in a system which will place more than one call to the
34 same subroutine at one time. Figs. 25a and 25e show a coded
35 call CCALL component and its circuit which can be used when
36 the subroutine is to be called from more than one location in
37 the state machine. The CCALL components are used in

1 conjunction with a SUBROUTINE component and possibly with a
2 type of join component such as the 2-BUS JOIN component shown
3 in Figs. 25b and 25f. Figs. 25c and 25g show the
4 SUBROUTINE component and its circuit.

5 Figs. 25e, 25f, and 25g are arranged to show how the
6 signal lines of these components may be connected in a
7 circuit if the three symbols show in Figs. 25a, 25b, and 25c
8 are used together. For simplicity, clock and clock enable
9 lines are not shown. In a system using these symbols, each
10 CCALL component which calls a particular subroutine will be
11 assigned a different code number. Thus the FORCE VALUE=@CODE
12 symbol 253 of CCALL circuit 2501 will have a different value
13 from each other CCALL circuit used in the same system with
14 the same SUBROUTINE component 2503. A high ACTIVE signal
15 applied to CCALL component 2501 causes AND bus 251 to apply
16 the value in cell 253 to its output terminal. This value is
17 applied through OR bus 256 of 2-BUS JOIN2 component 2502 to
18 the DATA_IN port of SRAM 258 in SUBROUTINE component 2503.
19 The ACTIVE signal applied to CCALL component 2501 is also
20 propagated through OR gate 257 of 2-BUS JOIN2 component 2502
21 to the ACTIVE line of SUBROUTINE component 2503, which
22 increments a counter 259 and pushes the DATA_IN value onto a
23 stack in SRAM 258. The ACTIVE signal is propagated by
24 SUBROUTINE component 2503 to the body of the subroutine, not
25 shown, which commences performing its task.

26 The designer must take care that a subroutine is not
27 called from one branch of a state machine while it is
28 responding to a call from another branch of the state
29 machine. If a second call to the subroutine occurs while the
30 body of the subroutine is still responding to the first call,
31 the code associated with the CCALL component invoking that
32 call of the subroutine will again be pushed onto the stack in
33 SRAM 258 as counter 259 is incremented. Since a stack is a
34 last-in-first-out device, the state machine will return to
35 the wrong branch of the state diagram. However, it is
36 acceptable for a subroutine to call itself, as discussed
37 below in connection with Fig. 26.

1 After the body of the subroutine has completed its
2 task, a high RETURN signal is propagated back through
3 SUBROUTINE component 2503 and 2-BUS JOIN2 component 2502 to
4 CCALL component 2501. The RETURN signal decrements COUNTER
5 259 in SUBROUTINE component 2503, and thus causes SRAM 258 of
6 SUBROUTINE component 2503 to remove the code of CCALL
7 component 2501 from the STACK and apply the value on its
8 DATA_OUT port through 2-BUS JOIN2 component 2502 to the B
9 input port of COMPARE circuit 252 of CCALL component 2501.
10 The value will be propagated through 2-BUS JOIN2 component
11 2502 to other components such as another CCALL component, not
12 shown, which will carry a different code. If the value at B
13 in CCALL component 2501 matches the value at A, a high signal
14 at the A_EQ_B terminal of COMPARE circuit 252 is applied to
15 an input of AND gate 255, which may be connected to another
16 STATE component, but in any case initiates further downstream
17 activity upon completion of the subroutine.

18 If the value at the B port of CCALL component 2501 does
19 not match the value at the A port, then the RETURN signal
20 did not apply to CCALL component 2501, and downstream
21 activity is not initiated by AND gate 255 of CCALL component
22 2501. That CCALL symbol with a matching code will activate
23 its NEXT flow line, as SUBROUTINE component 2503 removes the
24 value from its stack.

26 Example Recursive State Machine

27 Fig. 26 shows the state flow portion of a system which
28 uses the CCALL, 2-BUS JOIN2, and SUBROUTINE components
29 illustrated in Figs. 25a-25g.

30 After receiving a RESET signal the system moves from
31 START component 261 to STATE component 262, and at the next
32 clock cycle through JOIN2 component 263 and IF component 264
33 to either STATE component 265 or STOP component 269 depending
34 upon the high or low value of a signal on the TEST line of IF
35 component 264. If the TEST signal is high, at the next clock
36 signal, the system moves to STATE component 265, and at the
37 following clock signal to CCALL component 266. CCALL

1 component 266 has been assigned a code of 14. Therefore
2 CCALL component 266 applies the value 14 to a portion of its
3 SUB bus, which passes through 2-BUS JOIN2 component 270 to
4 SUBROUTINE component 271, which stores the value 14 on its
5 stack, and activates STATE component 272. As discussed
6 earlier, the ACTIVE line of STATE component 272 may be
7 connected to a logic function, which responds to the high
8 ACTIVE signal. At the next clock cycle, IF component 273 is
9 activated by STATE component 272. If the TEST line of IF
10 component 273 is FALSE, JOIN2 component 276 activates RETURN
11 component 277, which applies a high RETURN signal through
12 JOIN2 component 276 to IF component 273, and in turn to STATE
13 component 272, and SUBROUTINE component 271. As discussed in
14 connection with Fig. 25g, SUBROUTINE component 271 decrements
15 its counter 259 (See Fig. 25g) and applies the value in its
16 stack to its DATA_OUT port, which is propagated through 2-BUS
17 JOIN2 component 270 (Fig. 26) to CCALL component 266. Since
18 the value 14 matches the value stored in the stack of
19 SUBROUTINE component 271, CCALL component 266 applies a high
20 ACTIVE signal to its NEXT bus, thereby activating STATE
21 component 267. At the next clock cycle, STATE component 268
22 becomes active, after which the system cycles back through
23 JOIN2 component 263 to STATE component 265.

24 Note that RETURN component 277 also applies a high
25 RETURN signal through STATE component 275 to CCALL component
26 274. Therefore CCALL component 274 compares the value 14 on
27 the RETURN CODE portion of its SUB bus to its own CODE, which
28 is 10. Since these values do not match, CCALL component 274
29 does not send a signal on its NEXT bus to activate STATE
30 component 275.

31 If the TEST signal to IF component 273 is high at the
32 time STATE component 272 is active, IF component 273
33 activates (i.e., provides a high signal to) its TRUE bus in
34 response to an ACTIVE signal on the NEXT bus of STATE
35 component 272. This high signal on the TRUE bus in turn
36 activates CCALL component 274. CCALL component 274 sends a
37 code of 10 via 2-BUS JOIN2 component 270 to SUBROUTINE

1 component 271. Subroutine component 271 stores the value in
2 its stack, and the active state moves to STATE component 272.
3 Assuming the TEST signal to IF component 273 is still high,
4 CCALL component 274 sends another code of 10 via 2-BUS JOIN2
5 component 270 to SUBROUTINE component 271. Thus, the system
6 continues to cycle through the loop of components 274, 270,
7 271, 272, 273 until the TEST signal to IF component 273 goes
8 low. In this manner, several codes of 10 are stored in the
9 stack of SUBROUTINE component 271, the number of codes being
10 equal to the number of cycles through the loop.

11 When a low TEST signal is received by IF component 273,
12 the FALSE bus is activated, sending a signal through JOIN2
13 component 276 to RETURN component 277. The high signal
14 generated by RETURN component 277 propagates back through
15 JOIN2 component 276, IF component 273, and STATE component
16 272 to SUBROUTINE component 271, where this high RETURN
17 signal causes SUBROUTINE component 271 to place a code of 10
18 on its DATA_OUT line (see Fig. 25g), which is recognized as a
19 match by CCALL component 274. Thus STATE component 275 is
20 activated as SUBROUTINE 271 deletes one of the code 10 values
21 from its stack. The system thus cycles through STATE
22 component 275 for the number of times that the code 10 value
23 was added to the stack of SUBROUTINE 271. Finally, when
24 SUBROUTINE 271 places a value 14 on its RETURN CODE bus,
25 CCALL component 274 does not recognize the value 14 and thus
26 does not activate STATE 275; but CCALL component 266 does
27 recognize the value 14, and thus activates STATE component
28 267. The system continues to cycle until a low TEST signal
29 is received by IF component 264. When IF component 264
30 receives a low TEST signal, the system activates STOP
31 component 269, which ends activity of the state machine.

32 It can be seen that two different bus configurations are
33 used in the system of Fig. 26. A two-bus (or wide bus)
34 configuration connects the CCALL, 2-BUS JOIN2 and SUBROUTINE
35 components, while a bus configuration typically having five
36 bus lines such as described in connection with Figs. 13b,
37 14b, and 17b is used with the other components. Mixing of

1 the two bus configurations is internally consistent since the
2 CCALL, 2-BUS JOIN2 and SUBROUTINE components are connected
3 only to each other.

5 Complete Library Uses Several Bus Sizes

6 In a complete library, a set of basic symbols usable for
7 simple designs will be available. A further set of symbols
8 having more sophisticated features such as the invalid state
9 elimination feature will also be available for those desiring
10 the more sophisticated features. Another set usable for
11 branching state machines and reentrant designs will be
12 available for users desiring those features. Another set
13 usable with recursive state machines may also be available.

15 VECTOR Components

16 Even in a design using sophisticated symbols, it is
17 preferable to use the simple symbols where possible because
18 these require less hardware to implement. Included in a
19 library which includes these more sophisticated symbols is
20 preferably a VECTOR symbol which allows the bus size to
21 change, so the simple symbols usable in one part of a design
22 may be combined with more sophisticated symbols used in
23 another part of the design.

24 Figs. 27a and 27b show the symbol and related circuit
25 for a VECTOR1 component in which a single line is added to a
26 bus. Fig. 27c illustrates the circuit of Fig. 27b in
27 simplified form in combination with STATE3 and STATE4
28 components having a four-line bus width and a five-line bus
29 width, respectively.

30 Figs. 27d and 27e show the symbol and related circuit
31 for a VECTOR3 component and its circuit, which adds 3 lines
32 ELEM=A, ELEM=B and ELEM=C from one bus to another bus having
33 lines ELEM=0 ELEM=1, ELEM=2, and ELEM=3. The resulting NEXT
34 bus is a seven-line bus having lines ELEM=0 through ELEM=6.
35 VECTOR components of other widths are also provided in other
36 embodiments.

1 FORR and FORI Components

2 Another useful pair of components, related to the FOR
3 component, are a repeating FORR component and its related
4 FORI component. When several FOR components are needed, all
5 of which use the same count, it is possible to avoid having
6 separate counters for each FOR component. Instead, one FORR
7 component having one counter is used, and a plurality of FORI
8 component are linked to the FORR component. Thus only one
9 counter is needed to implement the count.

10 Figs. 28a and 28b show the FORR component, and Figs. 29a
11 and 29b show the FORI component. The FORR component of Fig.
12 28a and 28b includes one more bus than the FOR component of
13 Figs. 12a and 12b. The I_OUT bus communicates with FORI
14 components linked to the FORR component. In the particular
15 embodiment of Fig. 28b, a CYCLES bus receives a count value
16 from another part of the system. In another embodiment, the
17 user enters a number of cycles for the for-loop, as was
18 discussed and shown in Figs. 12a and 12b. Logic gates having
19 the same functions in Figs. 28b and 29b are given the same
20 reference numerals in the two drawings.

21 In operation, when multiple for-loops are to be used,
22 each with the same count value, one FORR component is used to
23 implement one for-loop, and all other for-loops using the
24 same count value are implemented by FORI components linked to
25 the FORR component counter through I_OUT and I_IN buses.
26 Specifically, the I_IN bus of one FORI component is connected
27 to the I_OUT bus of another FORI component or of the FORR
28 component.

29 As can be seen in Fig. 28b, provided the ELEM=1 line of
30 the PREV bus is providing a high CLK_EN signal, AND gate 319
31 passes the clock enable signal from OR gate 314 to the CLK_EN
32 terminal of counter 311. A high clock enable signal is
33 generated by OR gate 314 in response to a high ACTIVE signal
34 on the ELEM=3 line of the PREV bus, a high ACTIVE signal on
35 the ELEM=3 line of the BACK bus, or a high ELEM=1 signal on
36 the I_OUT bus (the clock enable signal from a FORI
37 component). OR gate 313 initiates loading of a new count

1 value from the CYCLES bus. The high LOAD signal from OR gate
2 313 is initiated by either a high ACTIVE signal from the
3 ELEM=3 line of the PREV bus or from the ELEM=0 line of the
4 I_OUT bus. In either case, the high LOAD signal causes
5 counter 311 to begin counting down for the number of clock
6 cycles indicated on the CYCLES bus.

7 If the count was initiated by a high ELEM=3 value on the
8 PREV bus of the FORR component, on the first clock signal
9 while the ELEM=3 line of the PREV bus is high, a high signal
10 from OR gate 315 causes AND gate 322 to generate a high
11 signal. This high signal is passed by OR gate 323 to the
12 ELEM=3 line of the LOOP bus. As discussed in connection with
13 Figs. 12a and 12b, the high signal on the ELEM=3 line of the
14 LOOP bus results in a high ELEM=3 signal on the BACK flow
15 bus. Thus, at the next clock cycle, a combination of the
16 high signal from OR gate 312 and the high output of OR gate
17 315 which is driven by a non-zero @CYCLES value, AND gate 321
18 provides a high value to OR gate 323. A high value is driven
19 by OR gate 323 until counter 311 reaches zero, whereupon OR
20 gate 312 provides a low signal. This low signal produces a
21 low output from AND gate 321, which in combination with a low
22 output from AND gate 322 produces a low ELEM=3 signal on the
23 LOOP bus. At this point AND gate 316 provides a high ELEM=3
24 signal on the NEXT flow bus to continue the operation.

25 If there is no high ELEM=3 signal on the PREV bus of the
26 FORR component, but there is a high ELEM=3 signal on the PREV
27 bus of a FORI component such as shown in Fig. 29b, AND gate
28 322 of Fig. 28b does not produce a high signal to initiate
29 the loop in FORR component of 28b. Instead the FORI
30 component having a high ELEM=3 signal initiates its own loop
31 with a high signal from its own AND gate 322. The high
32 ELEM=3 signal in the FORI component causes OR gate 313 of the
33 FORI component to provide a high ELEM=0 signal on its I_IN
34 bus. This high signal propagates through any other FORI
35 components in the chain to OR gate 313 of the FORR component,
36 where it initiates loading a count value. Since the BACK bus
37 of the FORR component is not providing a high ELEM=3 value to

1 AND gate 321 of the FORR component, this count value does not
2 activate the loop in the FORR component. However the high
3 output from OR gate 312 propagates back through the ELEM=2
4 line of the I_OUT bus of the FORR component and the ELEM=2
5 line of any intervening FORI components. This high signal is
6 applied to AND gates 321 of any intervening FORI components.
7 However, since the BACK buses of the intervening FORI
8 components are not providing a high signal to AND gates 321,
9 no response is initiated. In the FORI component which
10 initiated the count, the BACK bus is generating a high
11 signal, so in this component the loop remains active until
12 the count is done.

13 During the time the FORI component is active, its BACK
14 bus is providing a high signal to its OR gate 314. OR gate
15 314 propagates a high signal on the ELEM=1 line of the I_IN
16 bus through any other FORI components to the CLK_EN input of
17 counter 311, thus enabling the clock signal to operate
18 counter 311 of the FORR component. The ELEM=3 line of the
19 FORR component (and also any FORI components) carries a high
20 signal as long as the CYCLES value is not zero, and allows
21 the loop to be initiated.

22 When the count is done, a low signal output from OR gate
23 312 of the FORR component propagates through the ELEM=2 line
24 of the I_OUT bus, applying a low input to AND gate 321 of the
25 FORI component and of course to AND gate 321 of any other
26 FORI components. The ELEM=3 line of the LOOP bus of the FORI
27 component goes low. The low ELEM=2 line of the I_IN bus of
28 the FORI component produces a high inverted input to AND gate
29 316 of all FORI components. The FORI component which has
30 been active still carries a high signal on the ELEM=3 line of
31 its BACK bus. Thus AND gate 316 sends a high signal to OR
32 gate 318, which results in a high ELEM=3 signal on the NEXT
33 bus of the active FORI component. Thus the FORI component
34 has completed its task using the counter of the FORR
35 component without activating any other FORR or FORI
36 components in the chain. Moreover, only the single counter
37 311 was used, thus minimizing silicon area.

1 Note that because the FORR and FORI components in a
2 chain use the same counter, this implementation must not be
3 used when more than one component in the chain may be active
4 at one time.

6 Circuits which use only a single counter can be
7 provided for the PAUSE function, for example a PAUSER and
8 PAUSEI pair of library elements which use only one counter in
9 a chain.

11 Deleting of Unused Logic

12 Another option which may be used with the present
13 invention is to delete unused logic from a user's design
14 before implementing the design in programmable hardware. For
15 example, a JOIN3 component may be used to join only two
16 incoming buses. Or a state symbol having a return line may
17 be used where no return line is needed. For another example,
18 a VECTOR8 symbol may be included in a library and used to add
19 1 to 8 lines to a bus. Any unused lines would be deleted
20 before implementing the design in hardware. This logic
21 deleting allows a set of complex state machine components to
22 be implemented more simply when the needs are simple, thus
23 achieving efficient use of the hardware without requiring a
24 large library of symbols.

26 The above description is meant to be illustrative only,
27 and not limiting. For example, although the state flow
28 symbols shown in the figures have been given particular
29 appearances and particular names, other appearances and names
30 can alternatively be used. Further, symbols having
31 variations in the input and output terminals accessing the
32 symbols may be provided.

33 Further, although the present invention has been
34 described in considerable detail with reference to certain
35 state flow components which can be combined with logic
36 symbols, other embodiments are possible. For example, the
37 state flow components can be incorporated into a dedicated

1 CAE software program, instead of being added as a library to
2 an existing schematic capture package.

3 Further, additional state flow components may be defined
4 and employed. For example, the simple START symbol and
5 circuit may be replaced by a loop start component in which
6 activity is initiated either by a RESET signal as described
7 previously or by an output signal of another state component
8 applied to the D input of a flip flop. A component which
9 combines the lines and logic of the illustrated STATE
10 component with a VECTOR component may be provided.

11 Additional circuitry for eliminating invalid states can
12 be included. Other implementations than the one shown may
13 also be done. Logic for testing that a design will not
14 activate two states in a loop may be provided. Therefore,
15 the spirit and scope of the appended claims should not be
16 limited to the description of the versions contained herein.

17 Further, although the present invention has been
18 described in connection with field programmable logic array
19 chips such as available from Xilinx, Inc., the invention is
20 not limited to field programmable chips, or to logic array
21 chips. Mask programmable chips such as ASIC devices and even
22 custom chips may be designed using the structure and method
23 described herein.

1 CLAIMS

2 I claim:

3 1. A structure for configuring an integrated circuit chip
4 from a user's logic design comprising:

5 a computer having data entry means, display means and
6 computing means;

7 a schematic entry computer program comprising:

8 a library comprising symbols, said symbols including
9 a plurality of logic element symbols and at least
10 one state flow symbol;

11 means for receiving data entered through said data
12 entry means using said symbols; and

13 means for associating with each of said symbols a
14 circuit represented by that symbol and for
15 generating a list of hardware connections in
16 response to said data entered through said data
17 entry means.

19 2. A structure for configuring a logic array chip from a
20 user's logic design as in Claim 1 in which said at least one
21 state flow symbol comprises at least a STATE symbol having

22 a PREV input bus for receiving an output signal from
23 another state flow symbol;

24 a NEXT output bus for providing an input signal to
25 another state flow symbol;

26 an ACTIVE output terminal for indicating whether a state
27 represented by said STATE symbol is active.

29 3. A structure for configuring a logic array chip from a
30 user's logic design as in Claim 2 wherein said at least one
31 state flow symbol further comprises at least

32 an IF symbol having

33 a PREV bus for receiving at least one output signal
34 from another state flow component,

35 a TEST terminal for receiving a TEST signal,

36 a FALSE output bus for forwarding the signal from the
37 PREV bus when the TEST signal is low, and

1 a TRUE output bus for forwarding the signal from the
2 PREV bus when the TEST signal is high, and
3 a JOIN symbol having
4 at least two input buses for receiving input signals
5 from other state flow components and
6 an output bus for providing a high output signal when
7 an input signal on either of the input buses is
8 high.

10 4. A structure for configuring a logic array chip from a
11 user's logic design as in Claim 2 wherein said at least one
12 state flow symbol further comprises at least a START symbol
13 having
14 at least one input terminal for receiving a RESET
15 signal;
16 an ACTIVE output terminal for providing a high output
17 signal in response to said RESET signal;
18 a NEXT output bus for providing a high output signal in
19 response to said RESET signal.

21 5. A structure for configuring a logic array chip from a
22 user's logic design as in Claim 4 in which said START symbol
23 includes further input terminals for receiving CLOCK and
24 clock enable input signals, said ACTIVE output terminal and
25 said NEXT output bus providing said high output signals on
26 said ACTIVE output terminal and said NEXT output bus for the
27 length of one clock cycle if said clock enable signal is high
28 when said CLOCK signal goes high.

30 6. A structure for configuring a logic array chip from a
31 user's logic design as in Claim 2 wherein said at least one
32 state flow symbol further comprises at least
33 a FOR symbol comprising:
34 means for entering a number of cycles related to said
35 FOR symbol;
36 a PREV input bus for receiving an output signal from
37 another state flow symbol;

1 a NEXT output bus for providing an input signal to
2 another state flow symbol;
3 a LOOP output bus; and
4 a BACK input bus for receiving signals derived from
5 said LOOP output terminal and other circuitry which
6 performs a calculation selected by a user in
7 response to a signal on said LOOP output bus;
8 means for providing on said LOOP output bus a high
9 output signal in response to high signals on said
10 PREV or BACK input buses for a successive number of
11 times equal to said number of cycles.

13 7. A method for generating a netlist for a circuit design
14 comprising the steps of:
15 entering into a computer memory a first portion of the
16 circuit design using at least one state flow
17 component, each state flow component having an
18 associated flow diagram symbol;
19 entering into the computer memory a second portion of
20 the circuit design using at least one schematic
21 component linked by signal path segments, each
22 schematic component having an associated schematic
23 symbol;
24 entering into the computer memory a signal path segment
25 for transmitting a signal from one of the at least
26 one state flow component to one of the at least one
27 schematic component;
28 displaying the first portion on a video terminal as a
29 flow diagram, the second portion as a schematic
30 diagram and the signal path segment as a line
31 connecting the schematic diagram and the flow
32 diagram; and
33 generating a list of hardware units and connections
34 incorporating the first and second portions of the
35 circuit design and the signal path segment between
36 the state flow component and the schematic component.

- 1 8. A method of Claim 7 wherein the first portion of the
2 circuit design is a state machine and the state flow
3 component is a STATE component including
4 an input bus,
5 an output bus and
6 an ACTIVE terminal connected to the schematic
7 component,
8 the STATE component generating a high output signal at the
9 ACTIVE terminal when the state machine is in a state
10 associated with the STATE component.
- 12 9. A method of Claim 8 wherein the STATE component
13 comprises a flip-flop which
14 receives signals from the input bus including an
15 input signal line and a clock signal line,
16 transmits a high output signal from a Q output
17 terminal one cycle of the clock signal after
18 receiving a high signal on the input signal line,
19 the high output signal being transmitted to both
20 the schematic component and to a second state flow
21 component connected to the output bus.
- 23 10. A method of Claim 7 wherein the first portion of the
24 circuit design is a state machine and the state flow
25 component is an IF component having
26 a PREV terminal for receiving an output signal from
27 another state flow component,
28 a TEST terminal for receiving a TEST signal,
29 a FALSE output terminal for forwarding the signal from
30 the PREV terminal when the TEST signal is low, and
31 a TRUE output terminal for forwarding the signal from
32 the PREV terminal when the TEST signal is high.
- 34 11. A method of Claim 7 in which the first portion of the
35 circuit design is a state machine and the state flow
36 component is a JOIN component having
37 at least two input buses for receiving input signals

1 from other state flow components, and
2 an output bus for providing a high output signal when an
3 input signal on either of the input buses is high;
4 the JOIN component receiving on one of its input terminals
5 the output signal from the first STATE component.

7 12. A system for generating a netlist representing a circuit
8 design using a computer comprising:
9 means for storing schematic components in the computer,
10 the schematic components including
11 associated schematic symbols displayed on a
12 video terminal controlled by the computer, and
13 signal paths,
14 the schematic symbols and signal paths forming a
15 schematic diagram representation of a first portion
16 of the circuit design;
17 means for storing state flow components connectable by
18 flow bus lines to form a second portion of the
19 circuit design, the state flow components including
20 associated state flow symbols displayed on the
21 video terminal; and
22 flow bus lines;
23 said state flow symbols and flow bus lines forming a
24 state flow diagram representation of the second
25 portion of the circuit design; and
26 means for converting the first and second portions of
27 the circuit design into a netlist representation of
28 the circuit design.

30 13. A system for programming a programmable logic device to
31 implement a circuit design, comprising:
32 means for storing a first portion of the circuit design
33 in computer memory as state flow components connected
34 by flow bus lines, the state flow components
35 including state flow symbols which can be linked by
36 the flow bus lines and displayed on a computer video
37 terminal as a flow diagram;

1 means for storing a second portion of the circuit design
2 in computer memory as schematic components connected
3 by signal paths, the schematic components including
4 schematic symbols linked by the signal paths and
5 displayed on the computer video terminal as a
6 schematic diagram;
7 means for storing at least one path linking said first
8 and second portions; and
9 means for generating a netlist description of the
10 circuit design by combining the first portion and the
11 second portion.

13 14. A schematic capture package comprising:
14 a library of state flow components, each state flow
15 component including an underlying circuit design and
16 a state flow symbol;
17 means for forming a circuit diagram on a computer video
18 terminal in response to a user selecting state flow
19 components from said library and connecting said
20 components by flow bus segments; and
21 means for generating a netlist incorporating said
22 underlying circuit designs of each of said state flow
23 symbols as connected by said flow bus segments.

25 15. A schematic capture package as in Claim 14 in which
26 said library of state flow components further comprises
27 schematic components, each schematic component
28 including an underlying circuit design and a
29 schematic symbol;
30 said means for forming a circuit diagram on a computer
31 video terminal in response to a user selecting state
32 flow components further comprising means for forming
33 a circuit diagram on a computer video terminal in
34 response to a user selecting schematic components;
35 and
36 said means for generating a netlist incorporating said
37 underlying circuit designs of each of said state flow

1 symbols as connected by said flow bus segments
2 further incorporates said schematic components,
3 netlist including at least one connection between one
4 of said state flow components and one of said
5 schematic components.

7 16. A schematic capture package of Claim 14
8 wherein the schematic capture package includes a library
9 of schematic components, each schematic component
10 including an underlying circuit description and a
11 schematic symbol;
12 wherein selected schematic components can be connected
13 by signal path segments such that the schematic
14 symbols associated with the selected schematic
15 components and signal path segments are displayed on
16 the computer video terminal as a selected schematic
17 diagram representing a second portion of the circuit
18 design; and
19 wherein the means for generating incorporates the
20 underlying circuit design of the selected schematic
21 components into the netlist.

23 17. A schematic capture package of Claim 16 wherein
24 the first portion of the circuit design includes a first
25 state flow component having an output terminal,
26 the second portion of the circuit design includes a
27 first schematic component having an input terminal,
28 and
29 the first state flow component and the first schematic
30 component are connected by a first signal path
31 segment.

33 18. A schematic capture package of Claim 17 wherein the first
34 portion of the circuit design is a state machine and the
35 first state flow component is a STATE component including an
36 input terminal connected to a first flow bus segment, and an
37 output terminal connected to a second flow bus segment and

1 the first signal path segment, and wherein the STATE
2 component generates a high output signal at the output
3 terminal when the state machine is in an active state
4 associated with the STATE component.

6 19. A schematic capture package of Claim 18 wherein the
7 STATE component further comprises a D flip-flop receiving
8 signals from the first flow bus segment including an input
9 signal line and a clock signal line, the D flip-flop
10 transmitting the high output signal from a Q output terminal
11 for one cycle of the clock signal after receiving a high
12 input signal, the high output signal being transmitted to
13 both the schematic component and to a second state flow
14 component connected to the second flow bus segment.

16 20. A schematic capture package of Claim 19 wherein the
17 second state flow component is an IF component and the
18 circuit design further includes a second STATE component
19 connected to the IF component by a third flow bus segment, a
20 third STATE component connected to the IF component by a
21 fourth flow bus segment, and a second schematic component
22 connected to the IF component by a signal line, the IF
23 component receiving the output signal from the first STATE
24 component and a TEST signal from the second schematic
25 component; wherein the IF component transmits a high signal
26 to an input terminal of the second STATE component if the
27 TEST signal is high and the output signal from the first
28 STATE component is high, and the IF component transmits a
29 high signal to an input terminal of the third STATE component
30 if the TEST signal is low and the output signal from the
31 first STATE component is high.

33 21. A method for generating a netlist for a circuit design
34 comprising the steps of:
35 entering into a computer memory a circuit design using a
36 plurality of state flow components, each state flow component
37 having an associated flow diagram symbol;

- 1 entering into the computer memory a signal path segment
- 2 connecting an output of one of the state flow components to
- 3 inputs of at least two other state flow components; and
- 4 generating a netlist incorporating the state flow
- 5 components and the signal path segment.

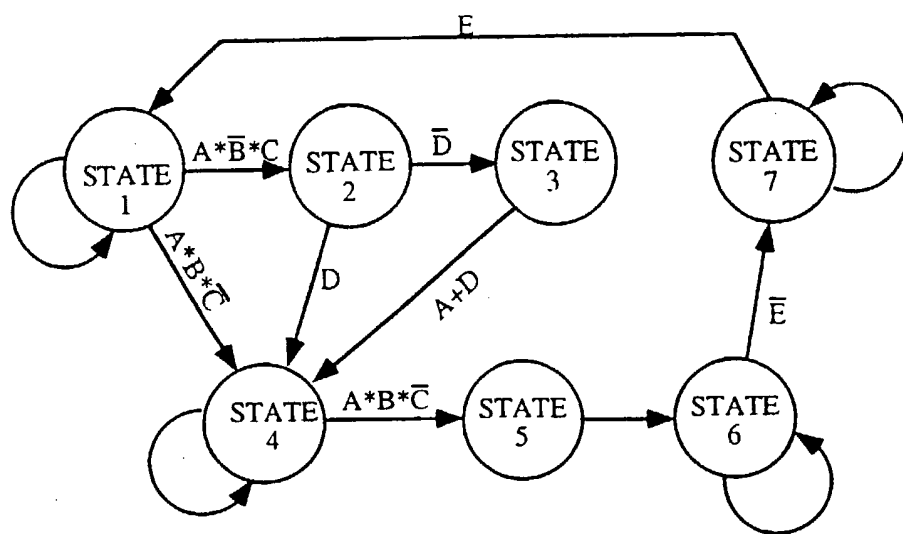


Fig. 1
Prior Art

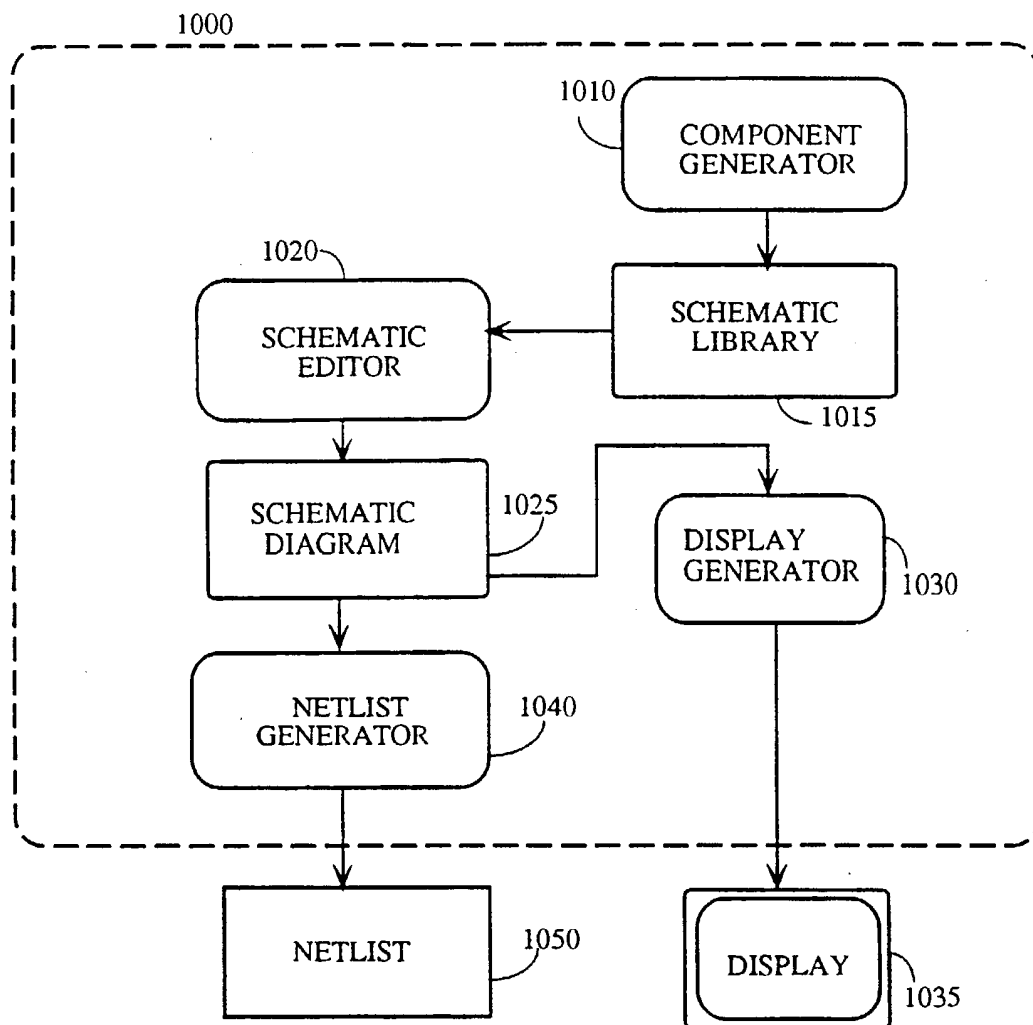
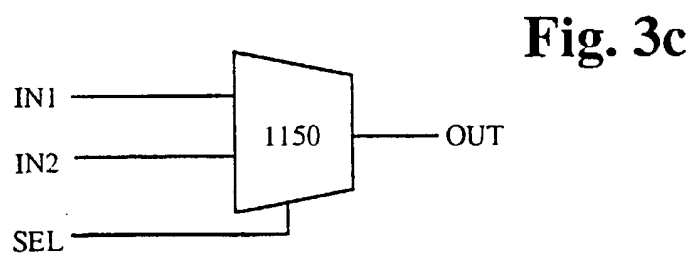
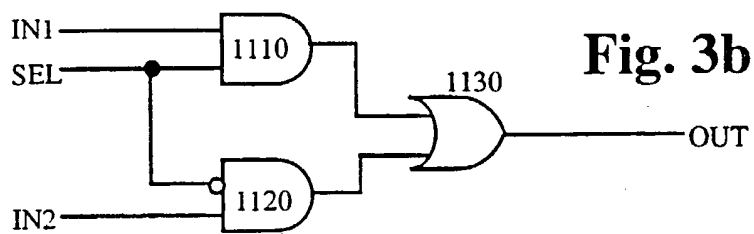
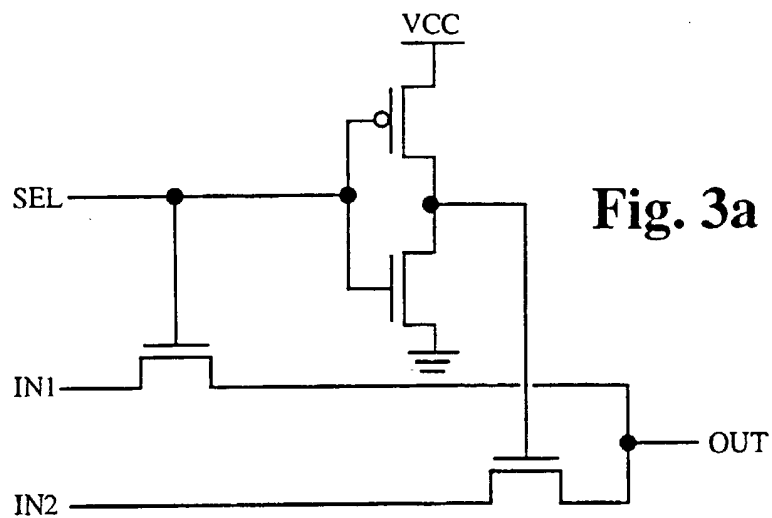
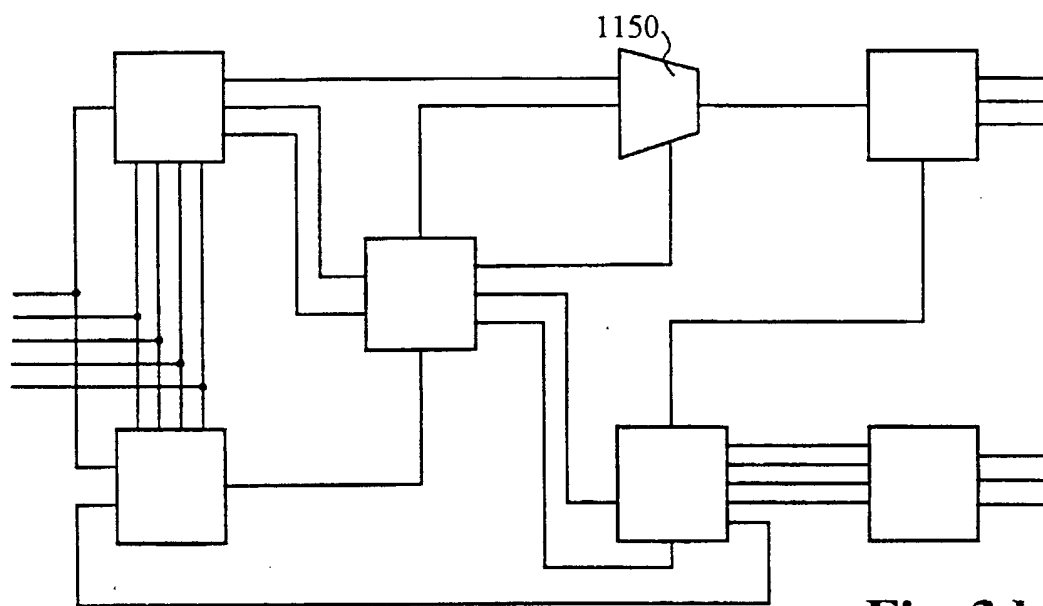
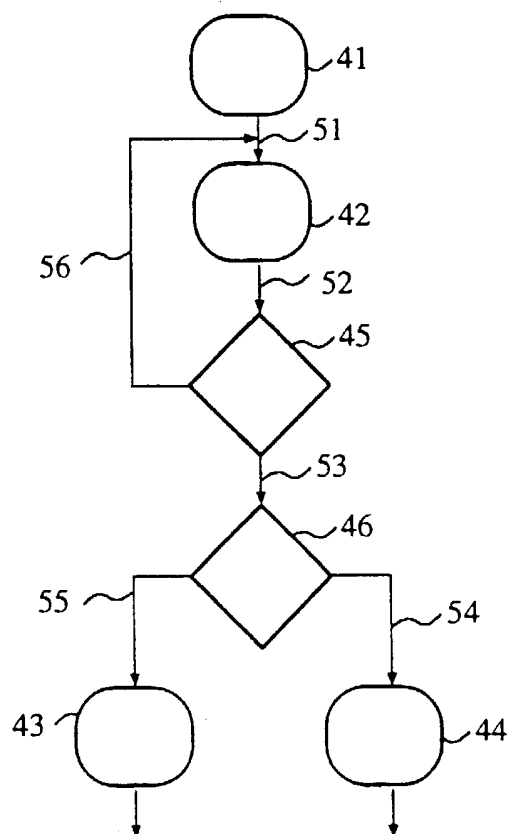
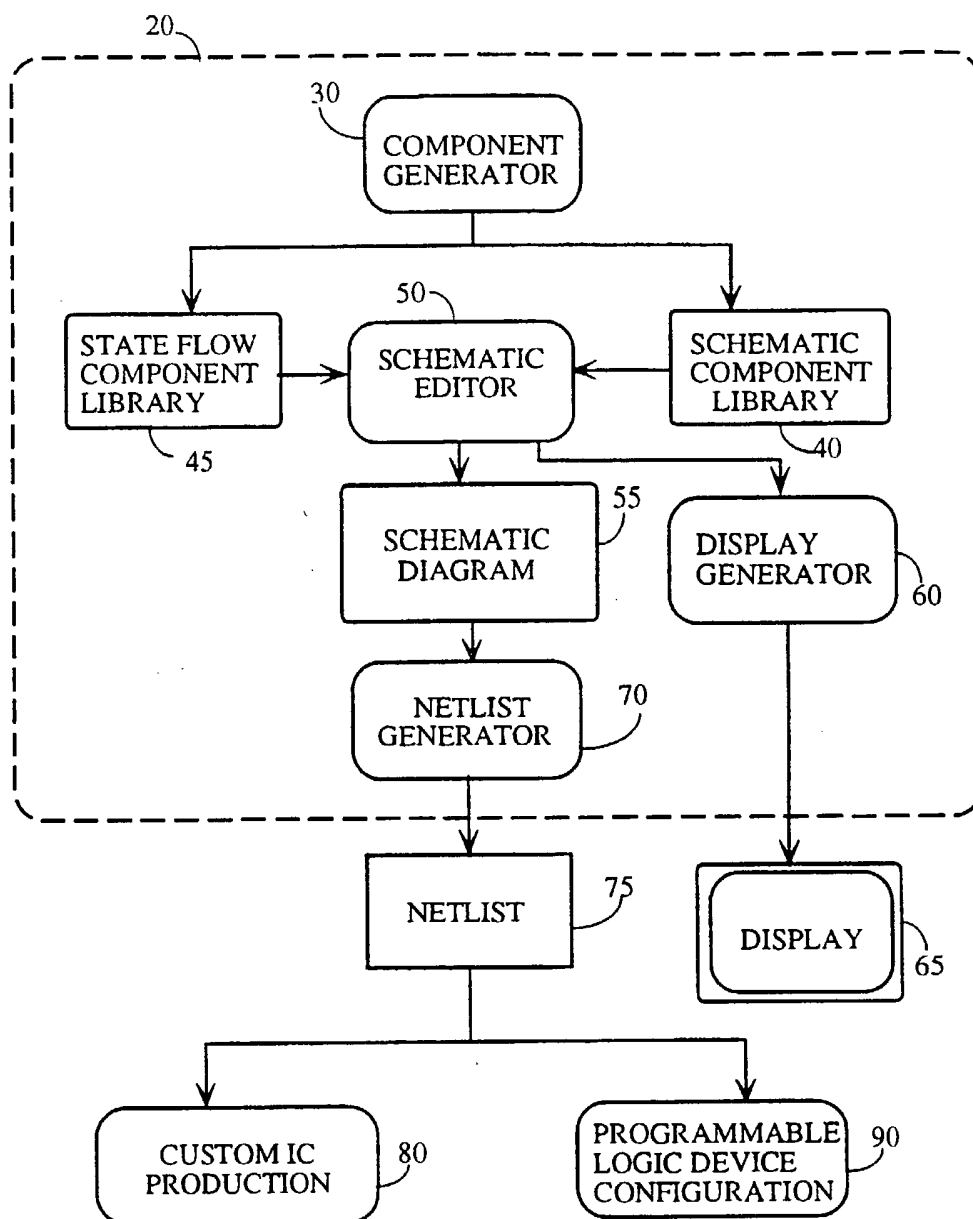


Fig. 2
Prior Art



**Fig. 3d****Fig. 4**

**Fig. 5**

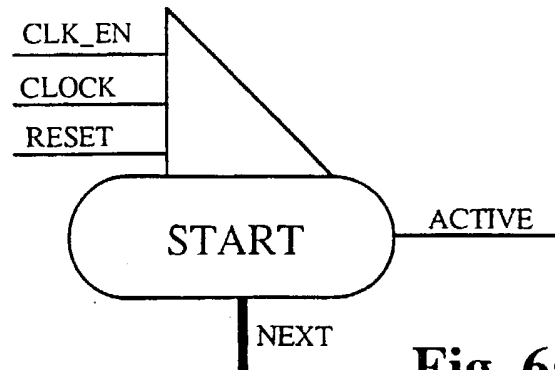


Fig. 6a

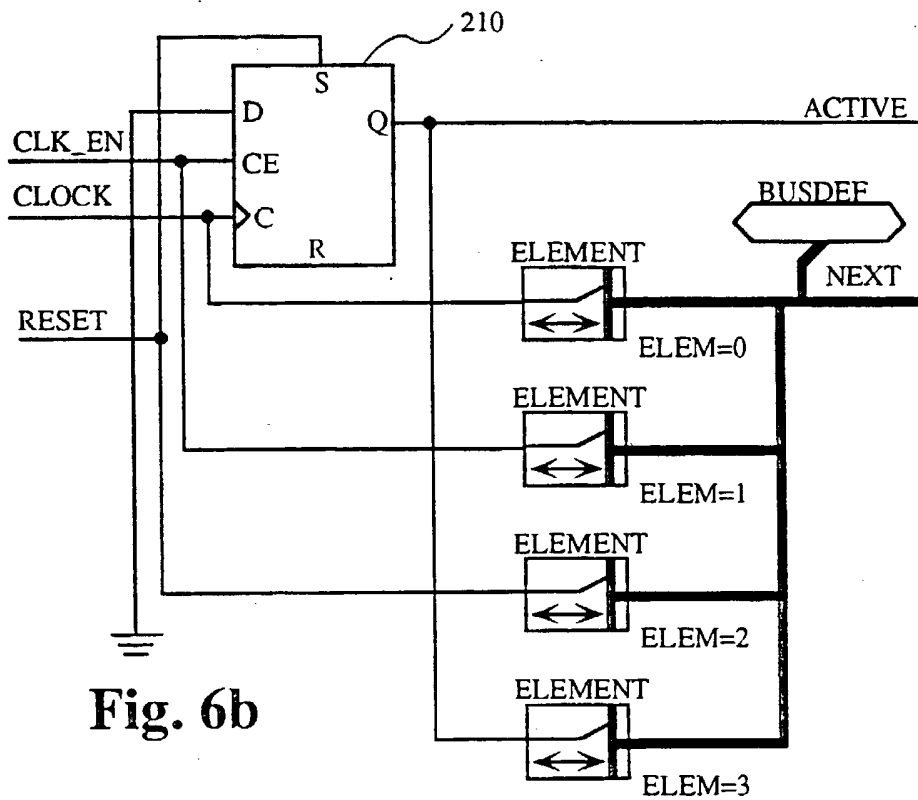


Fig. 6b

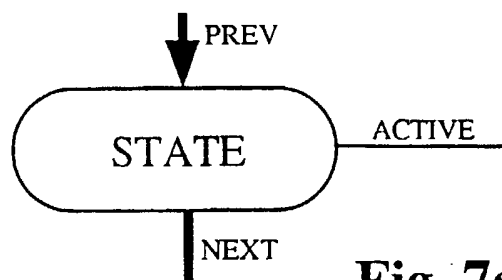


Fig. 7a

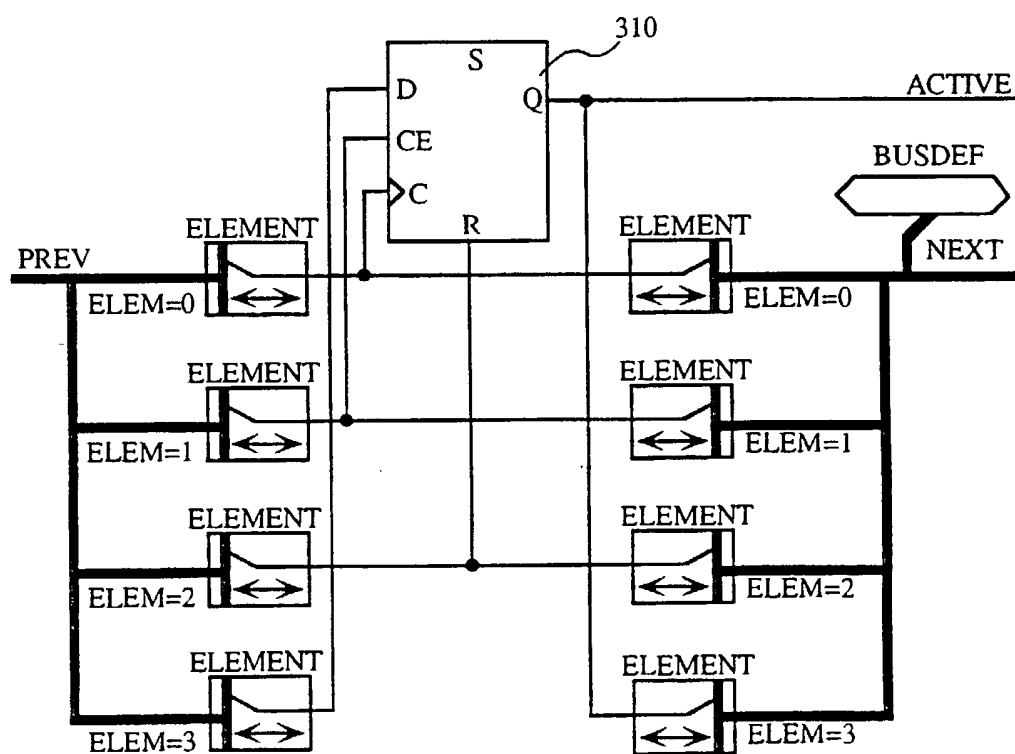


Fig. 7b

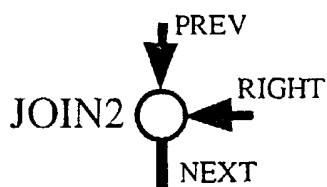


Fig. 8a

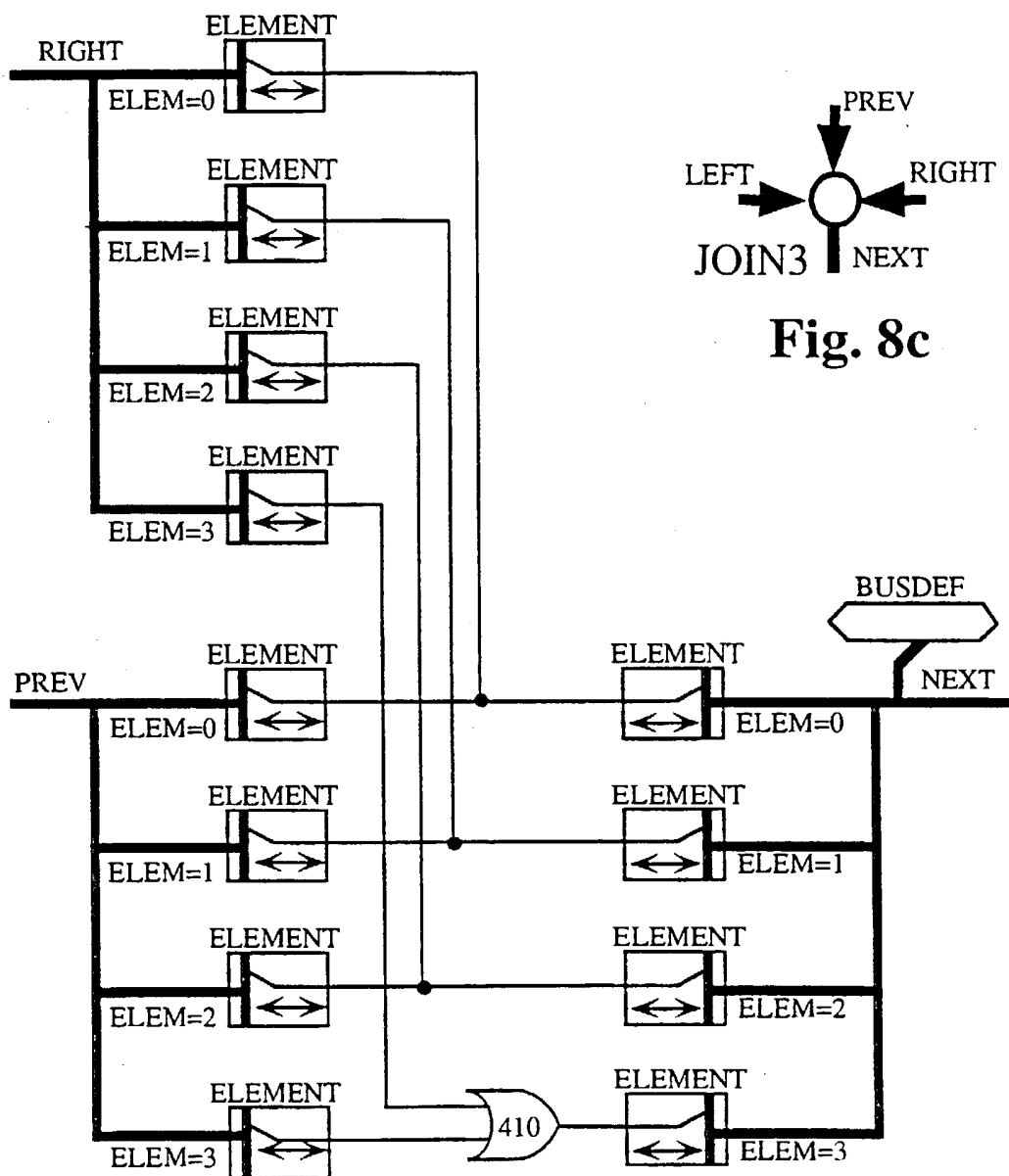


Fig. 8b

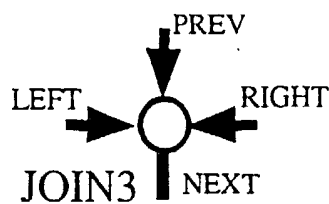


Fig. 8c

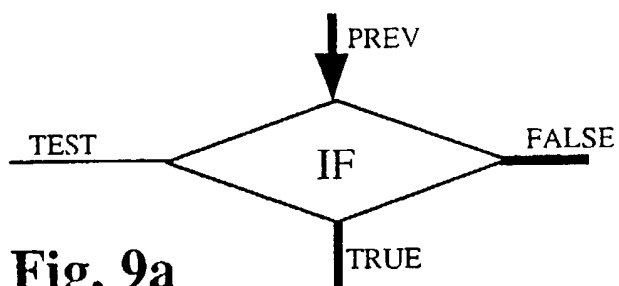


Fig. 9a

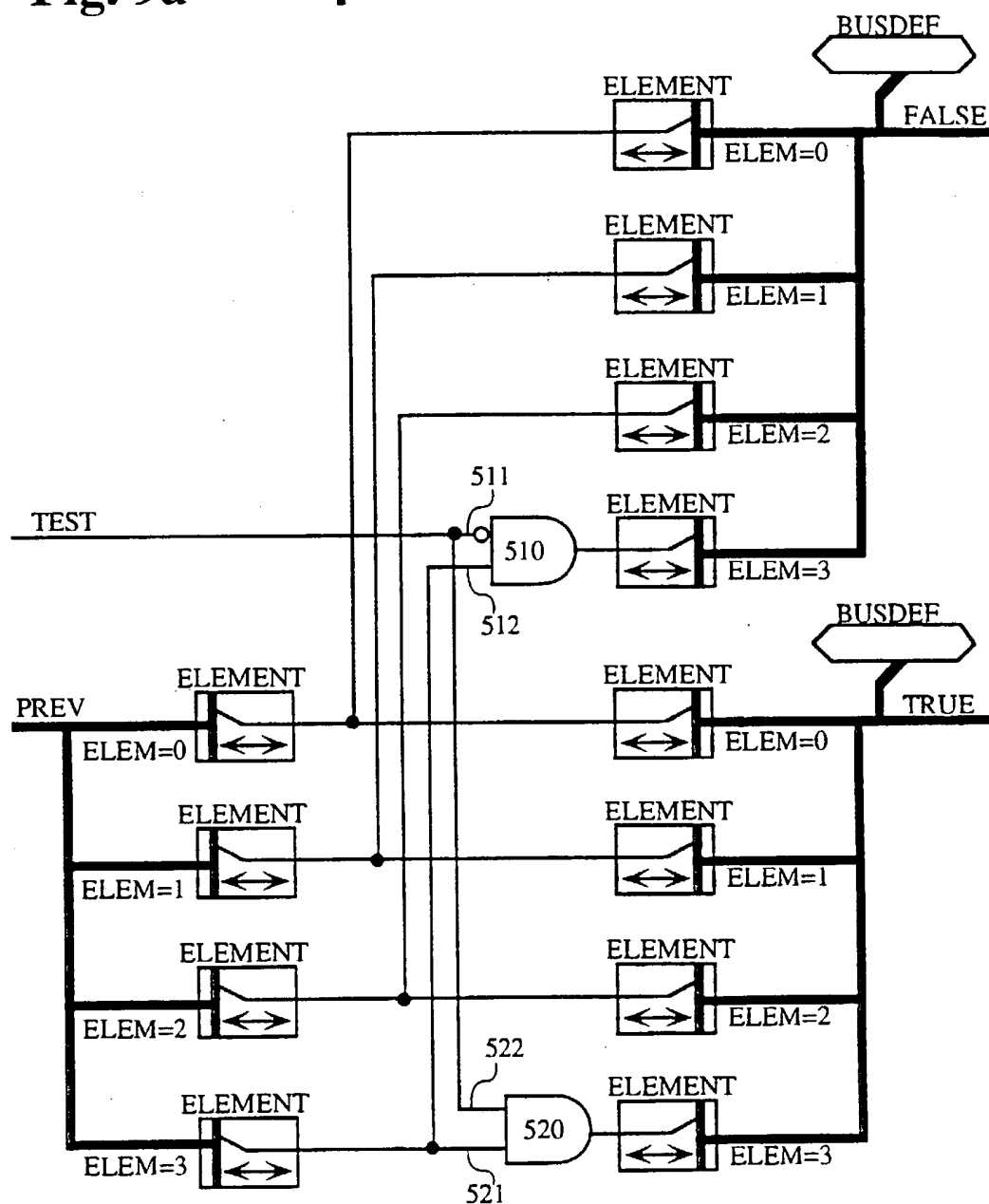


Fig. 9b

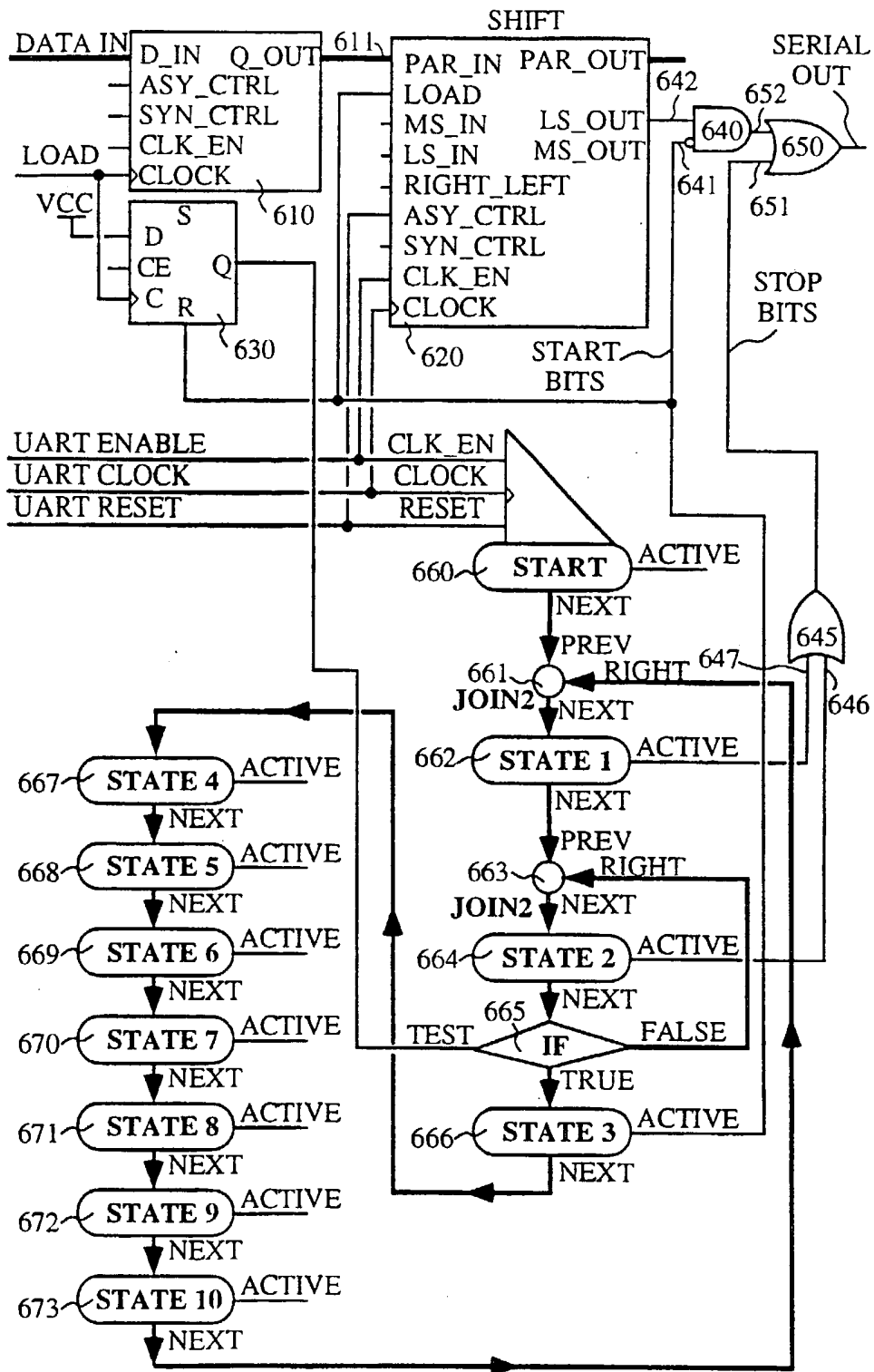
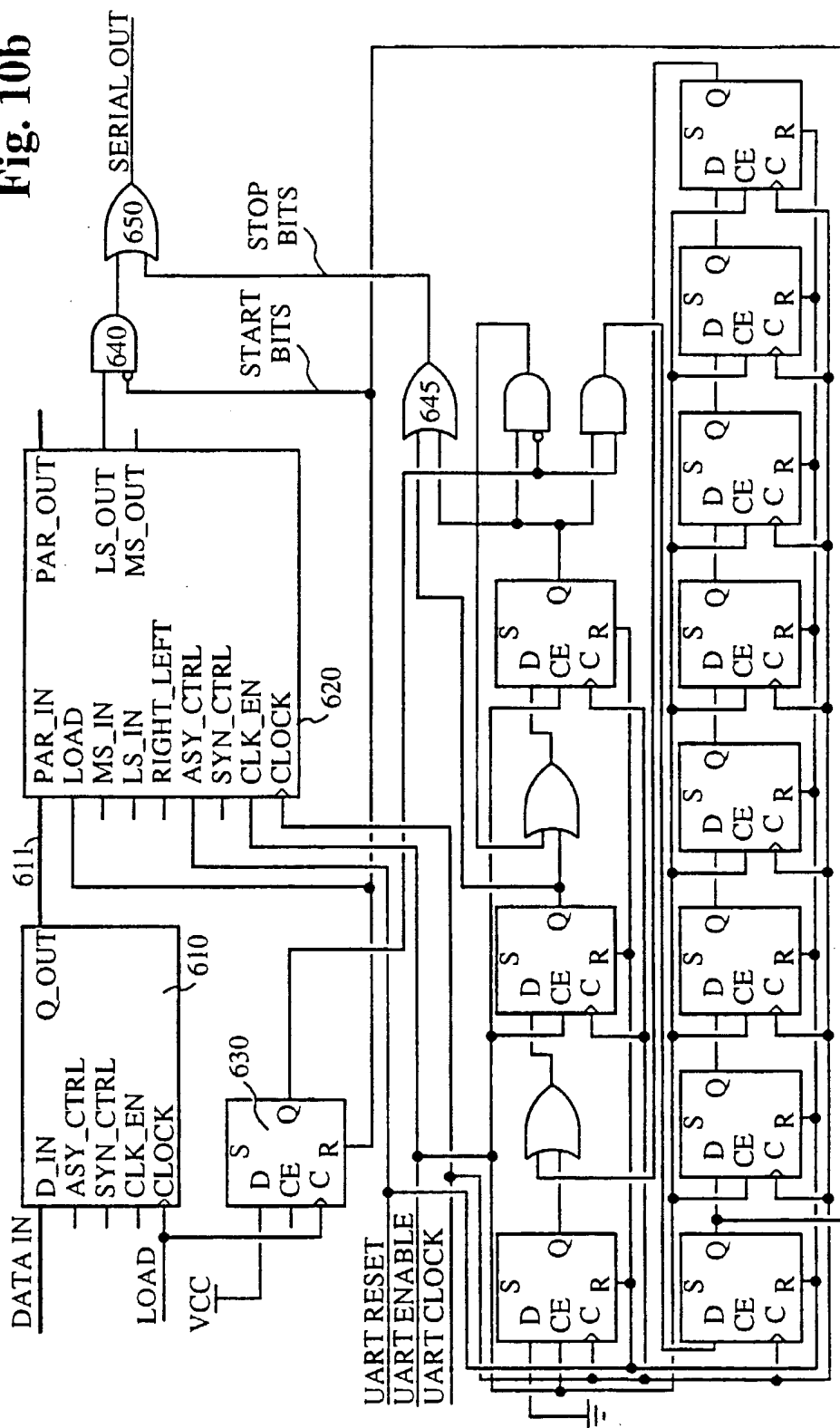


Fig. 10a

Fig. 10b



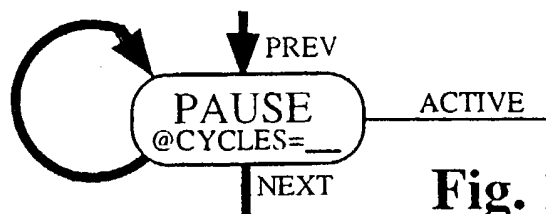


Fig. 11a

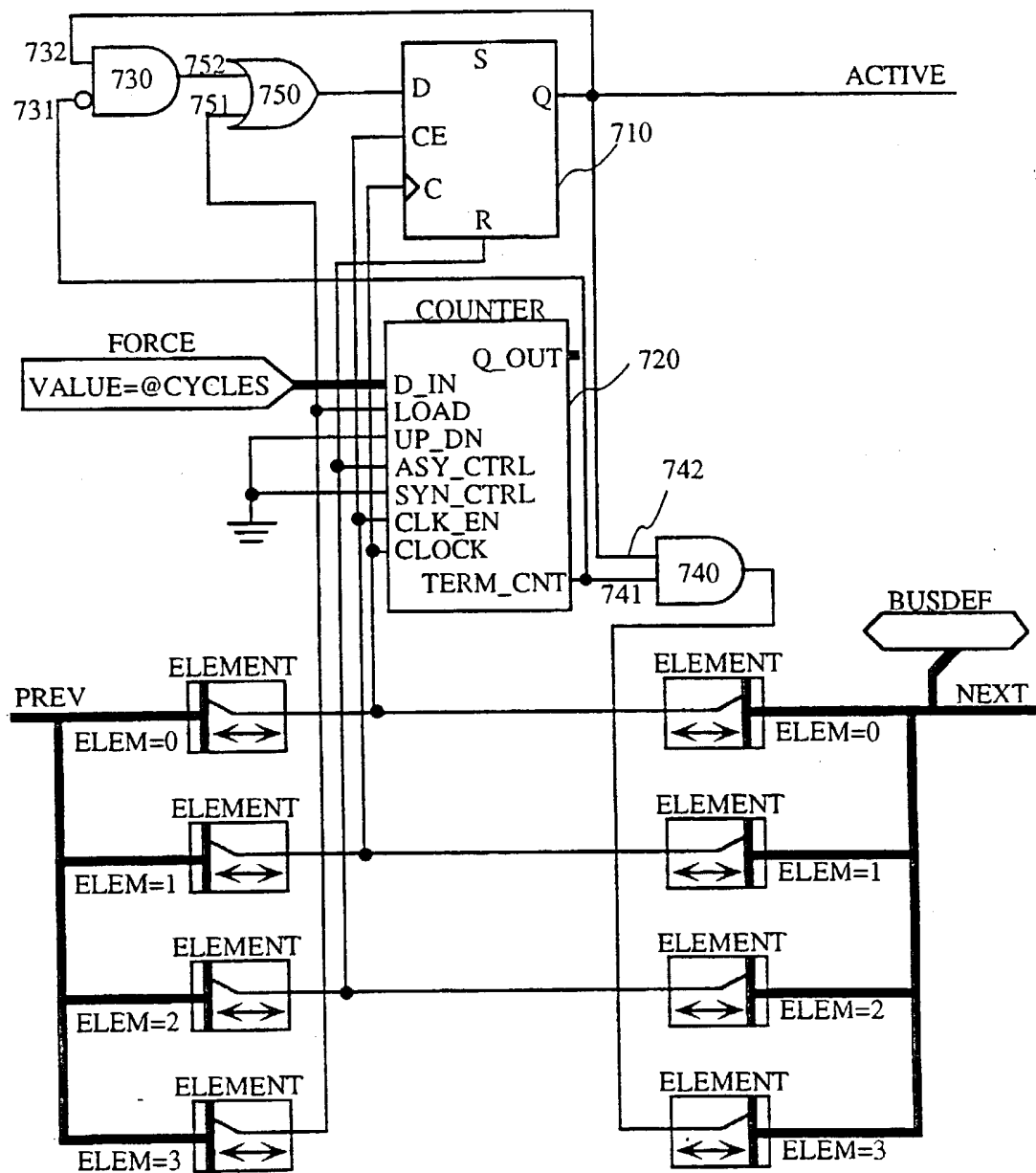


Fig. 11b

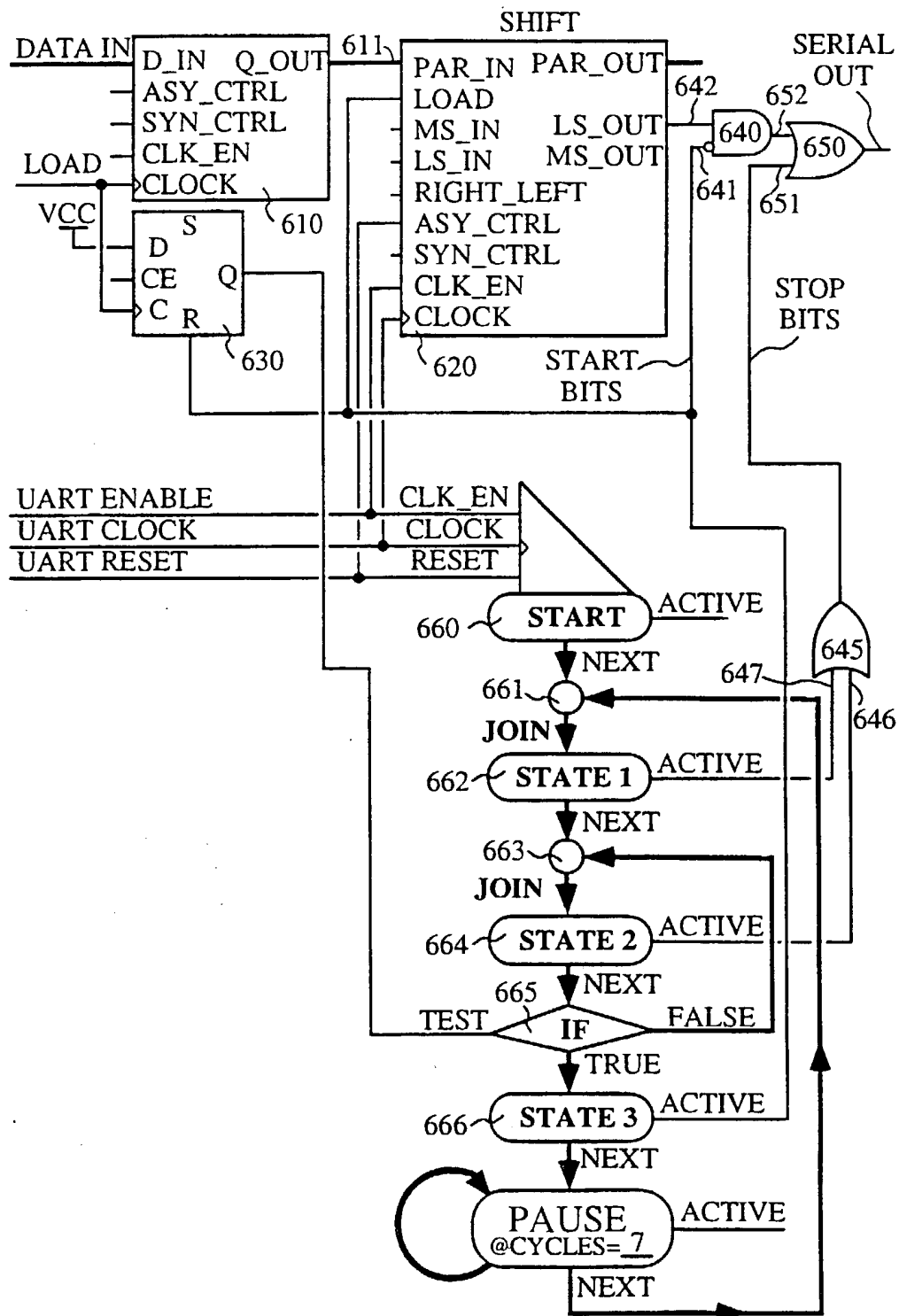


Fig. 11c

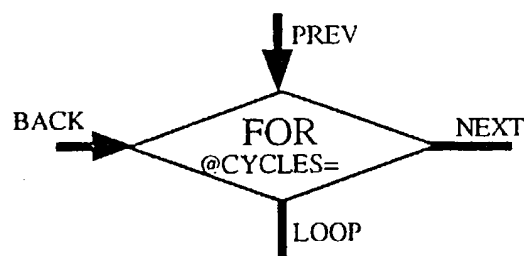


Fig. 12a

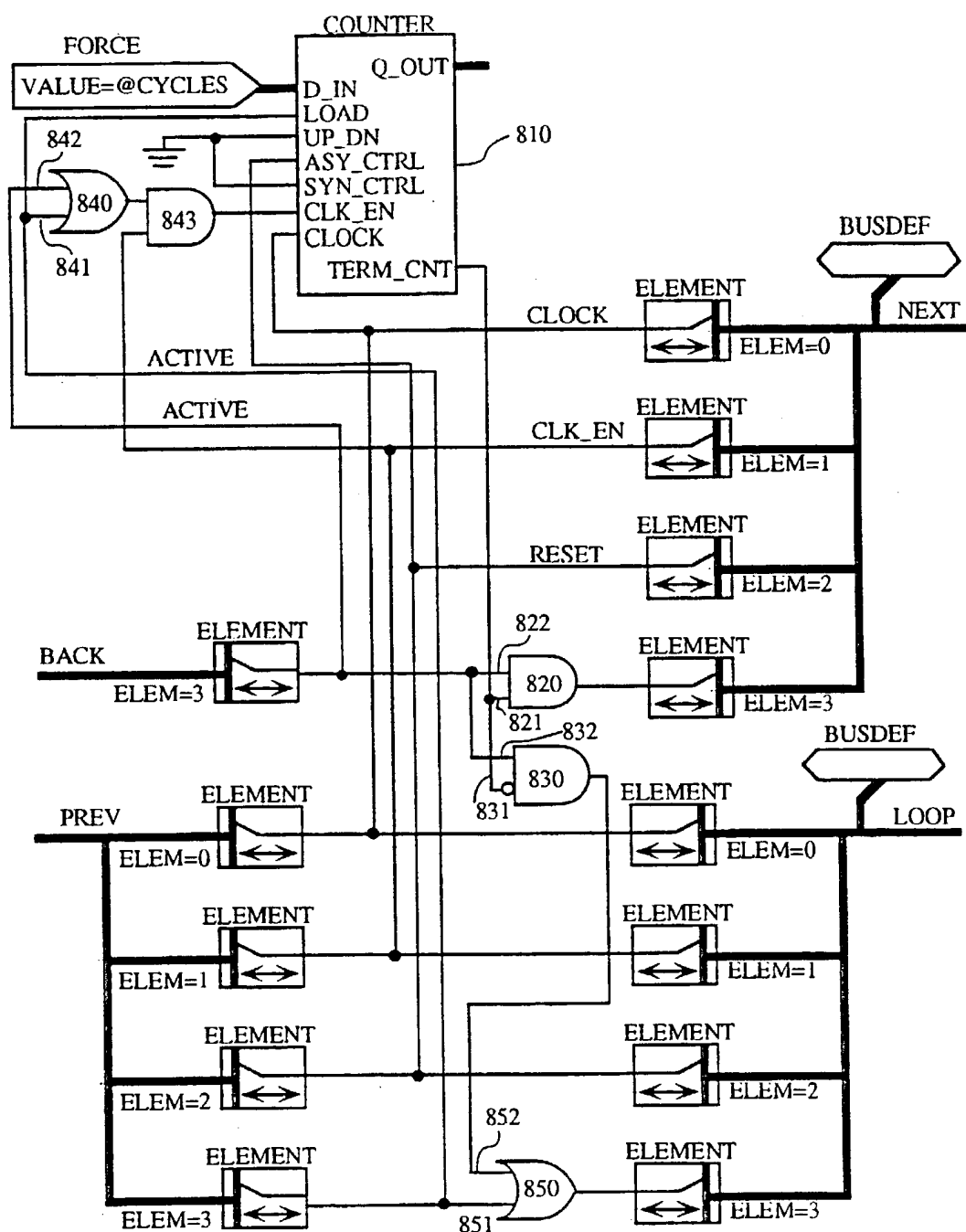


Fig. 12b

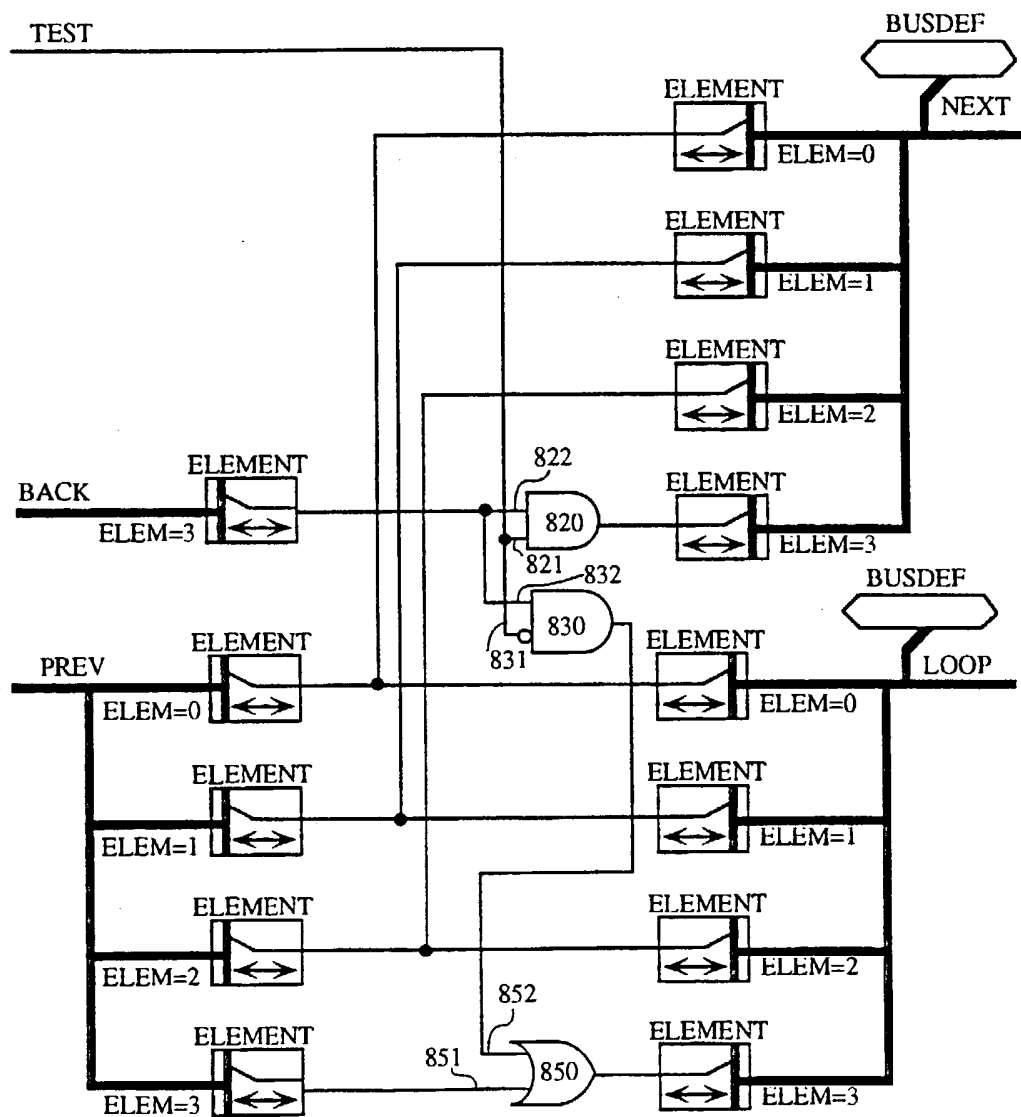
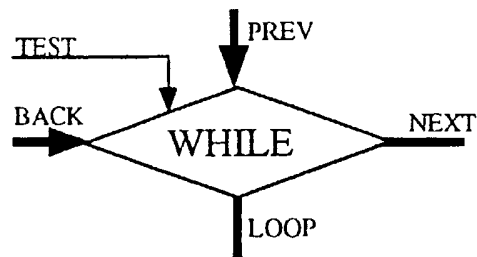


Fig. 13a

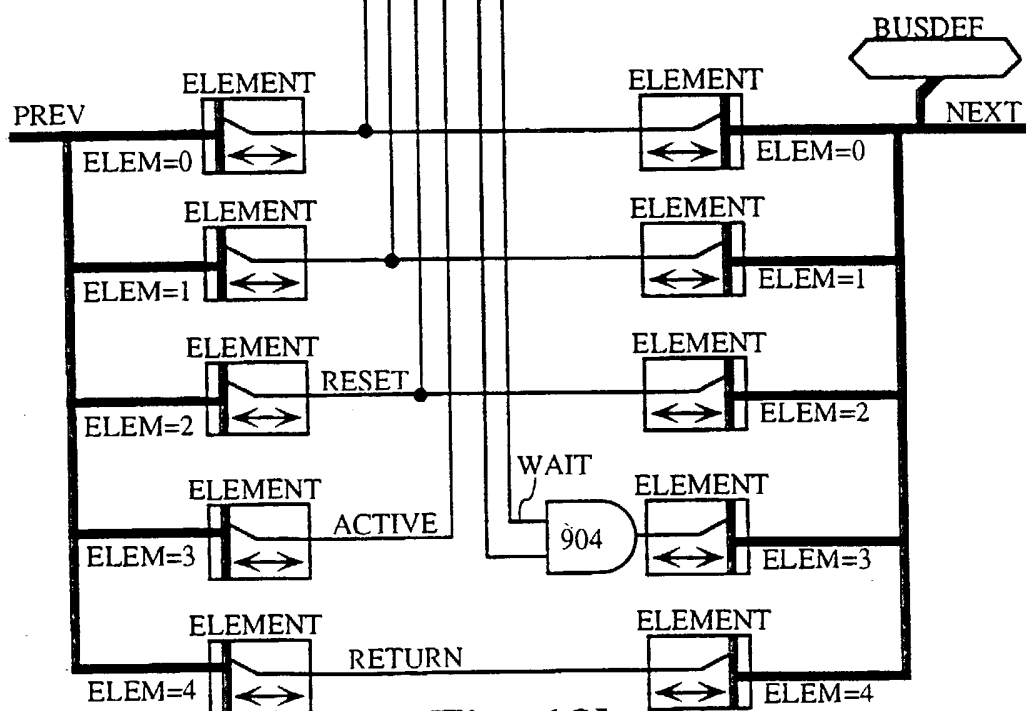
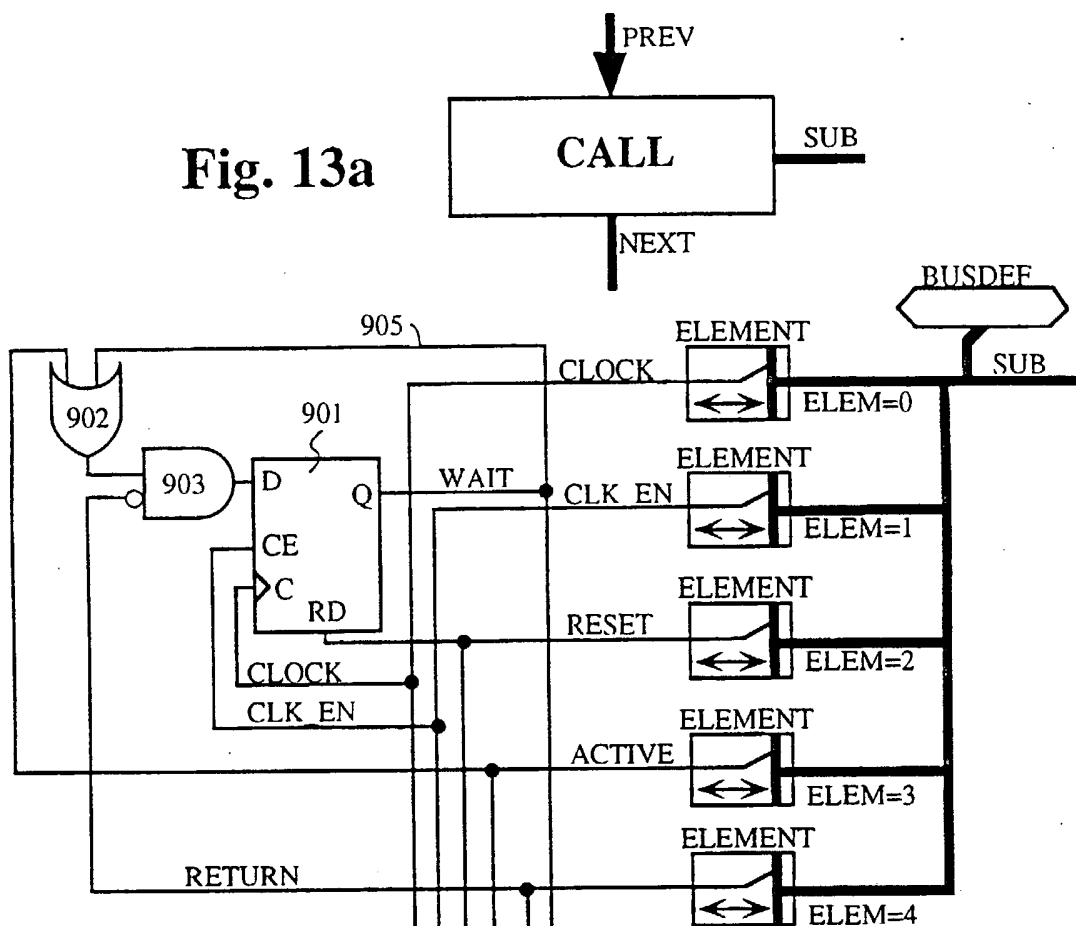


Fig. 13b

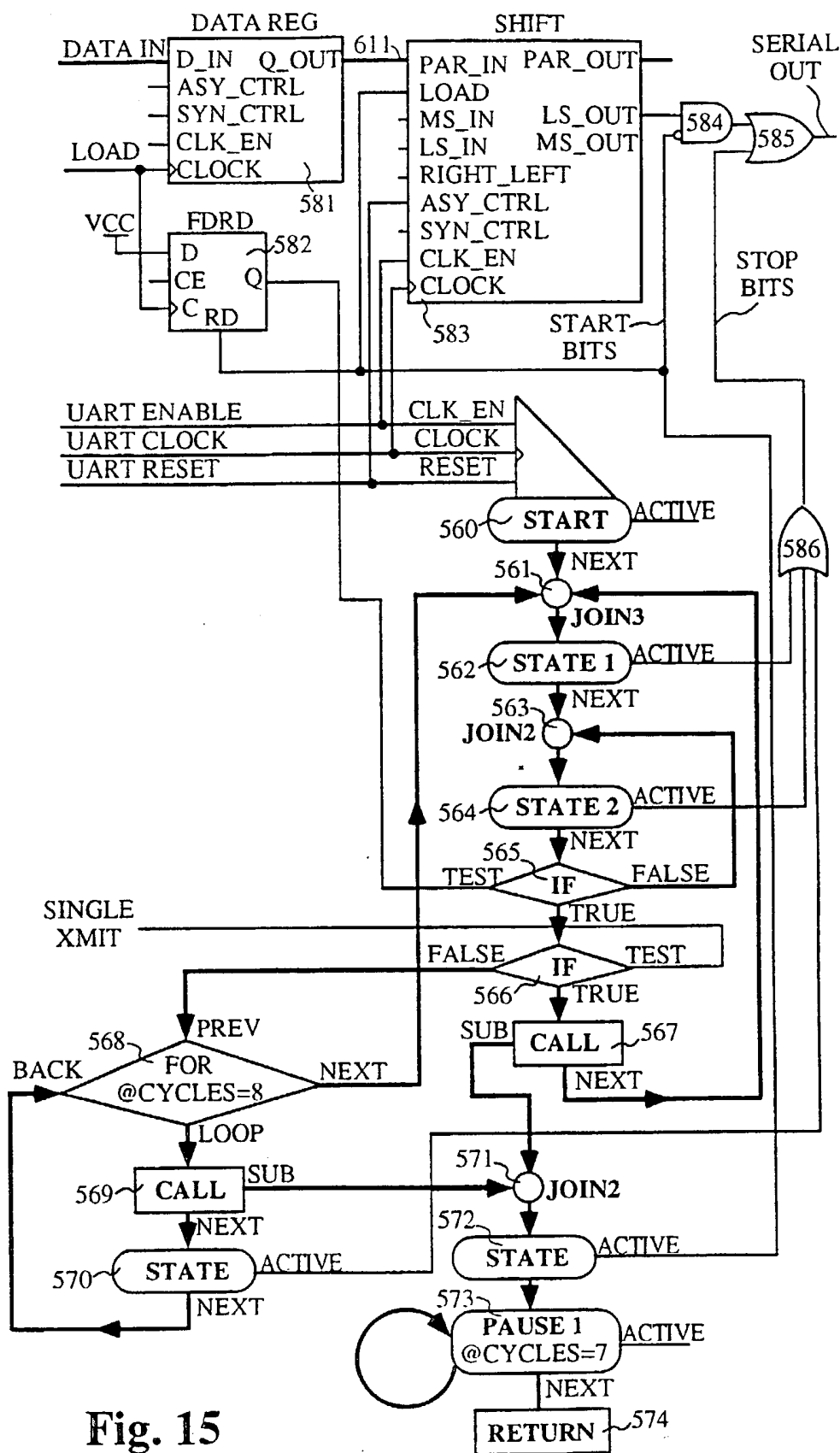


Fig. 15

Fig. 14a

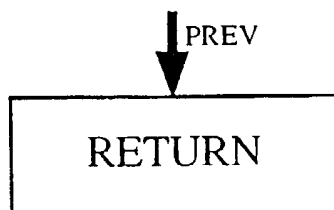


Fig. 14b

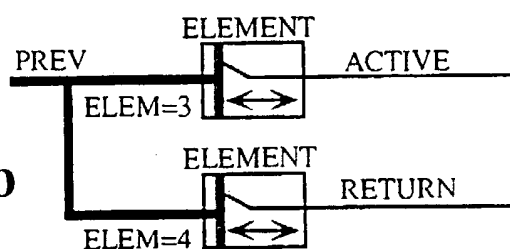


Fig. 16a

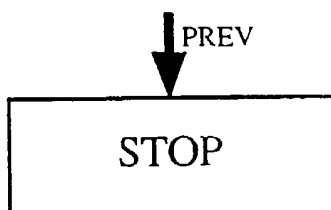
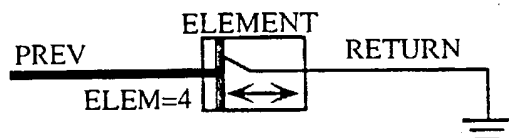


Fig. 16b



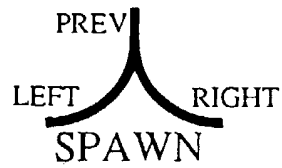


Fig. 17a

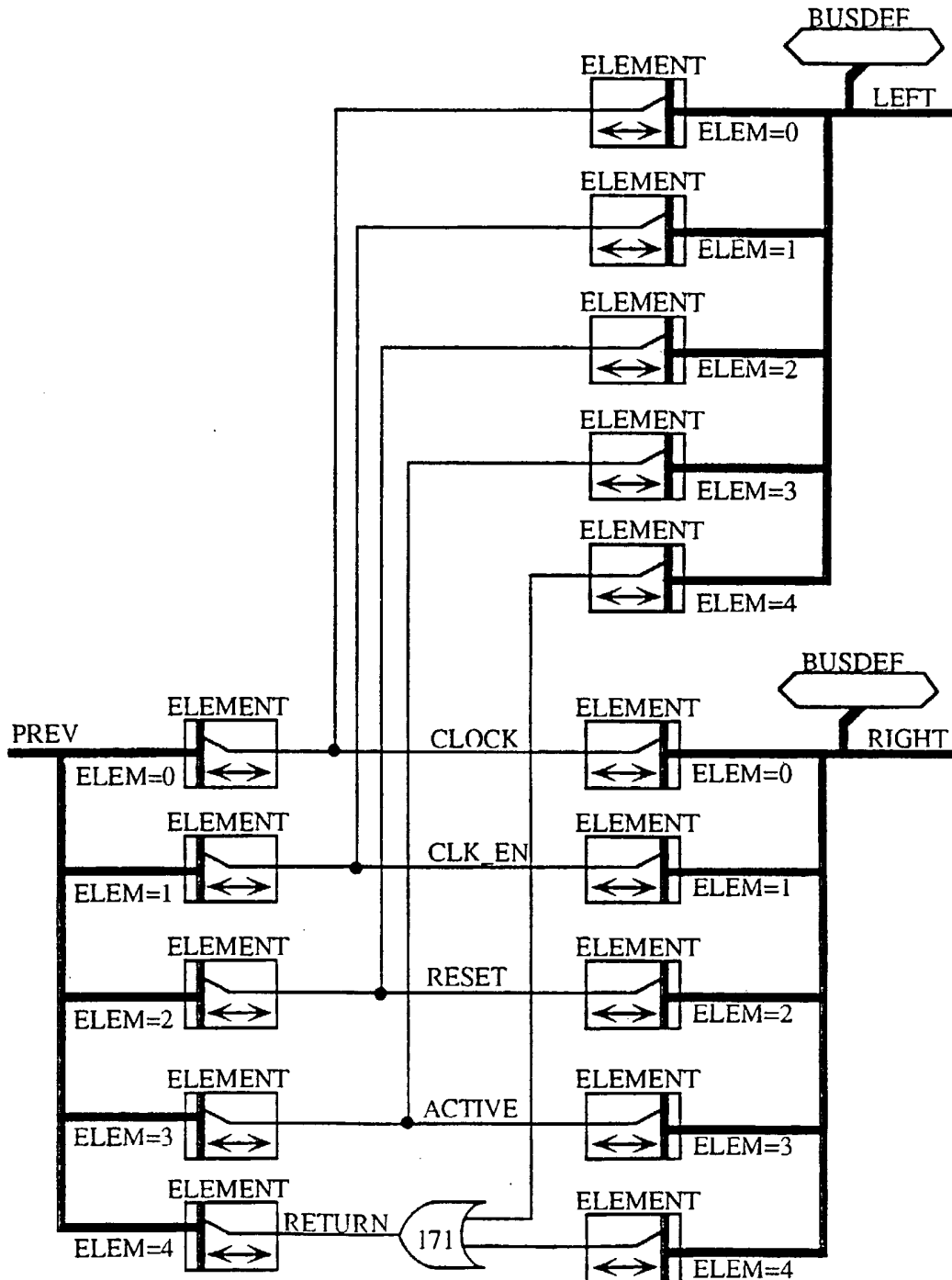


Fig. 17b

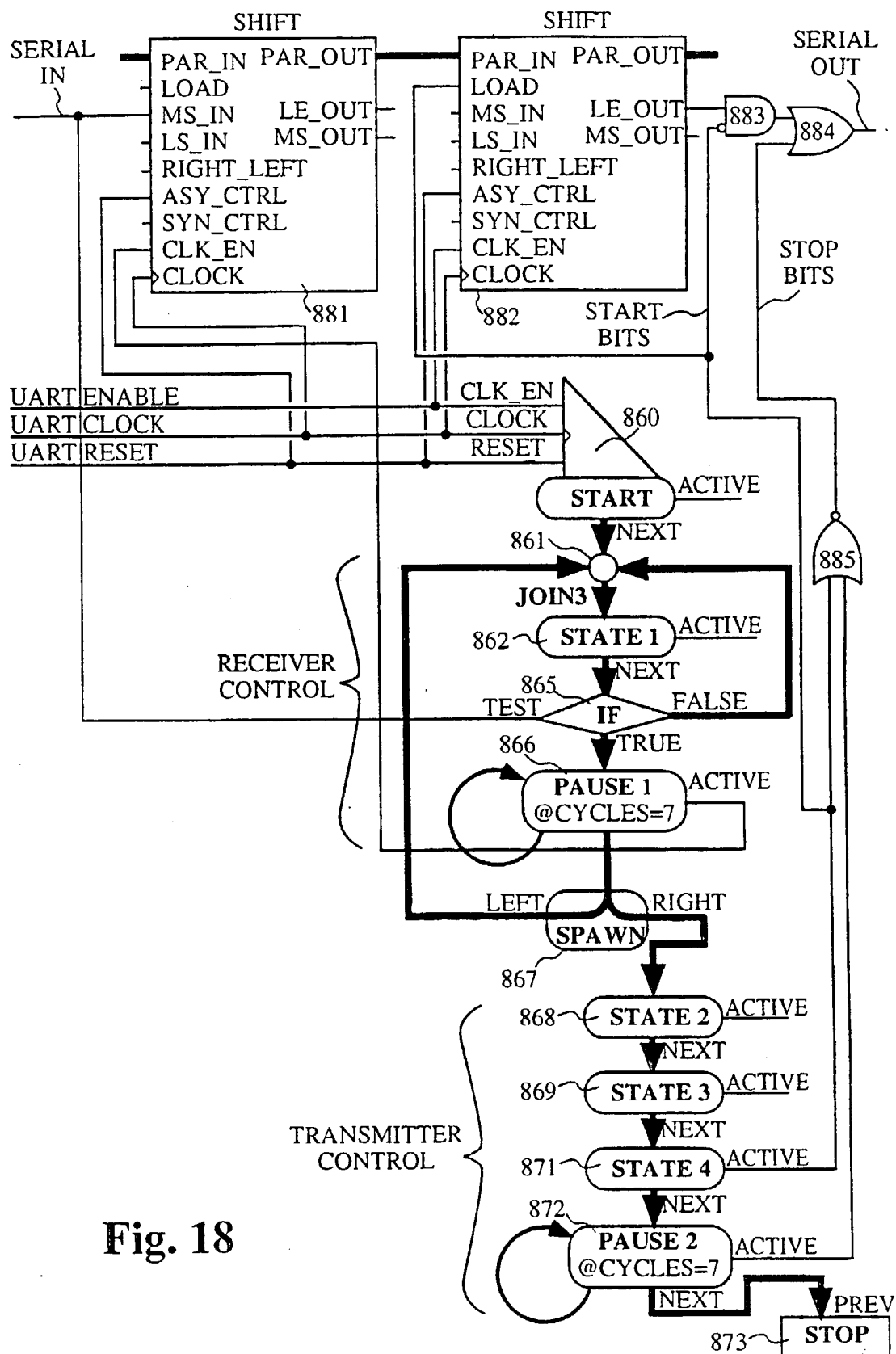




Fig. 19a

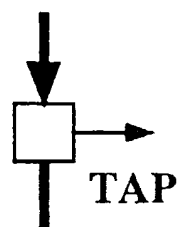


Fig. 19c

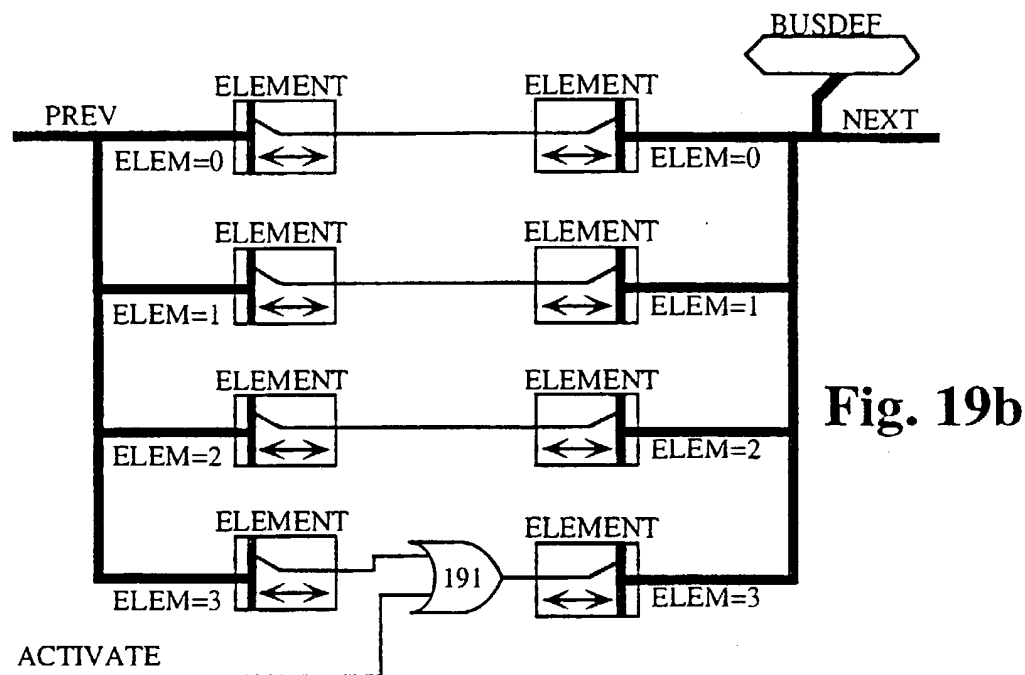


Fig. 19b

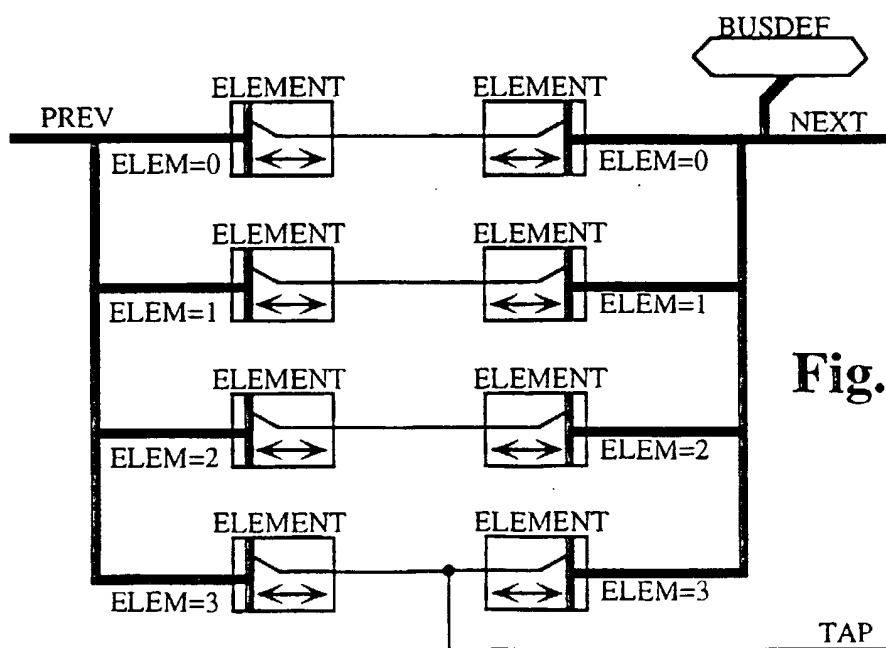
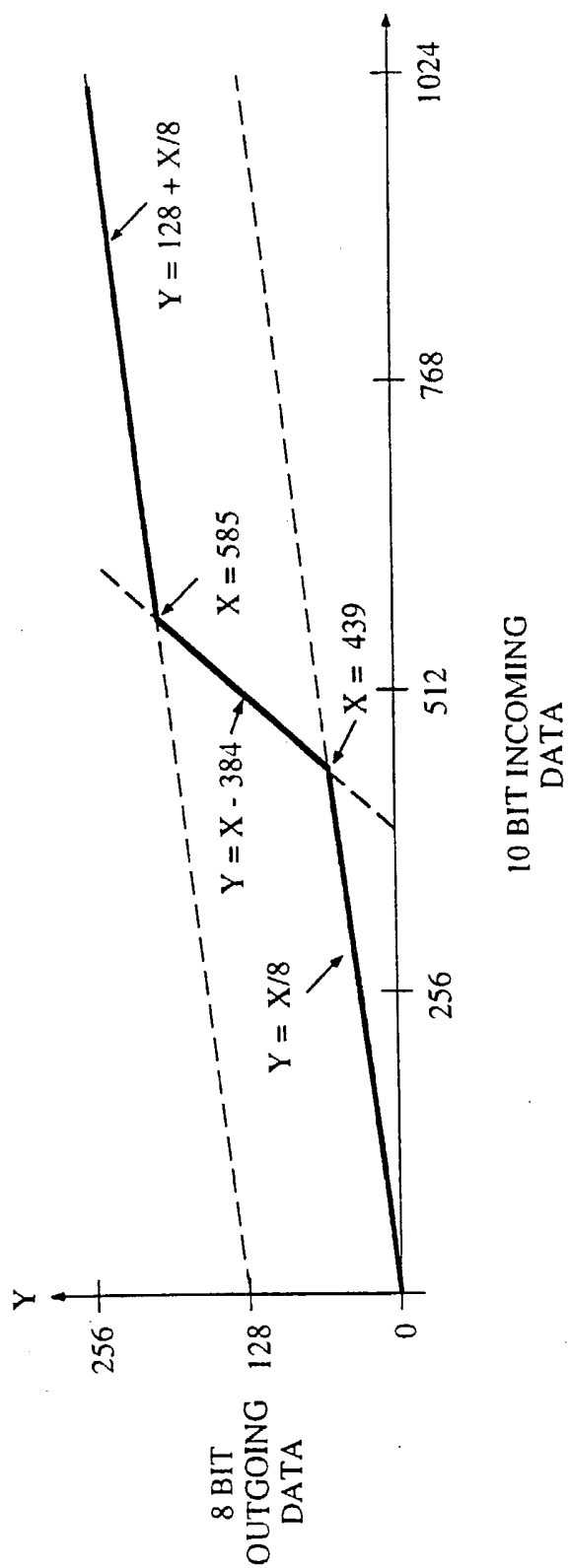


Fig. 19d

**Fig. 20**

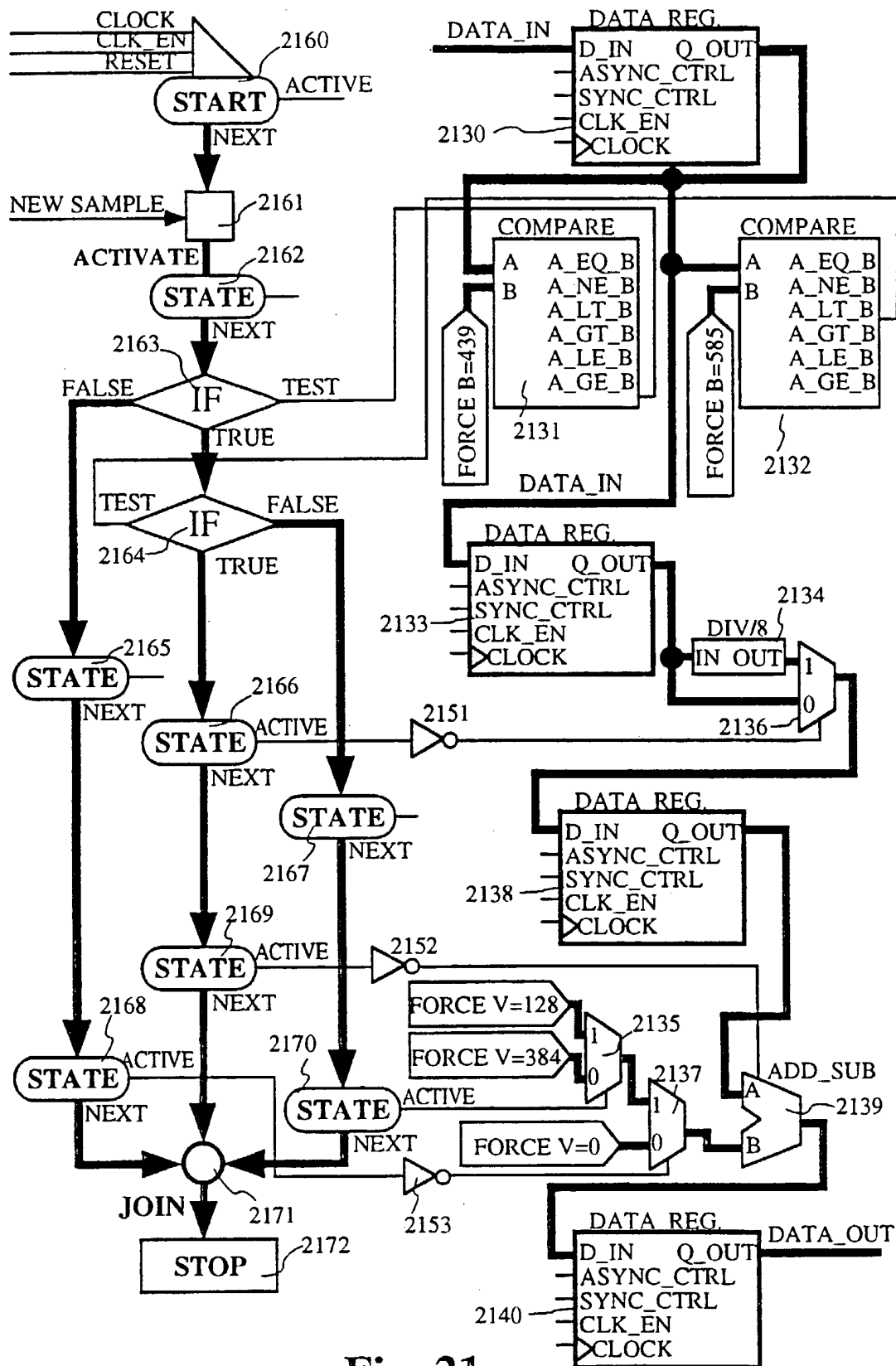


Fig. 21

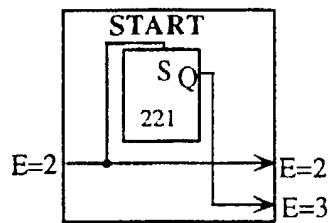


Fig. 22a

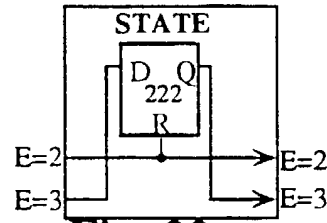


Fig. 22c

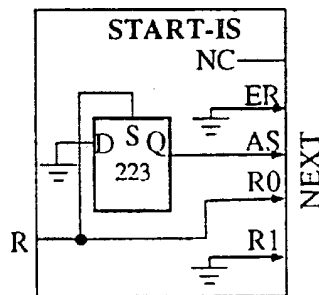


Fig. 22b

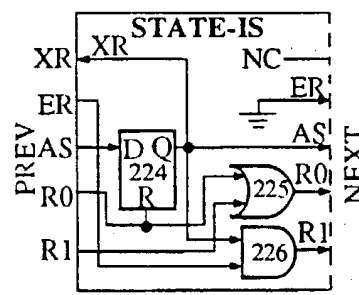


Fig. 22d

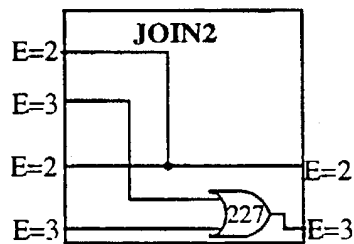


Fig. 22e

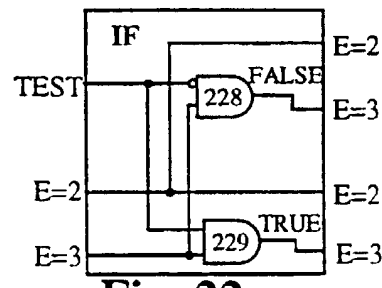


Fig. 22g

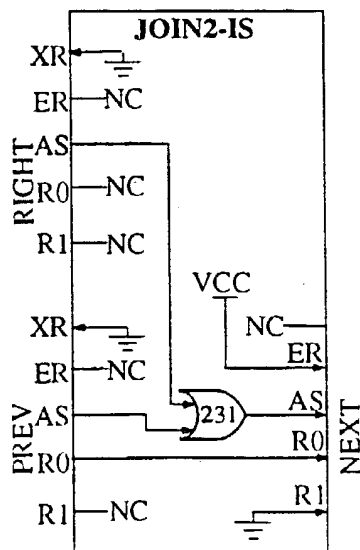


Fig. 22f

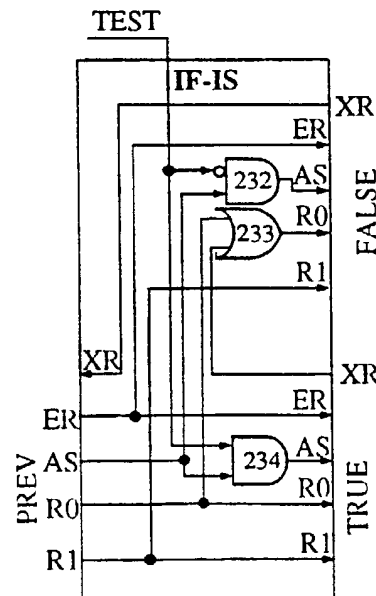


Fig. 22h

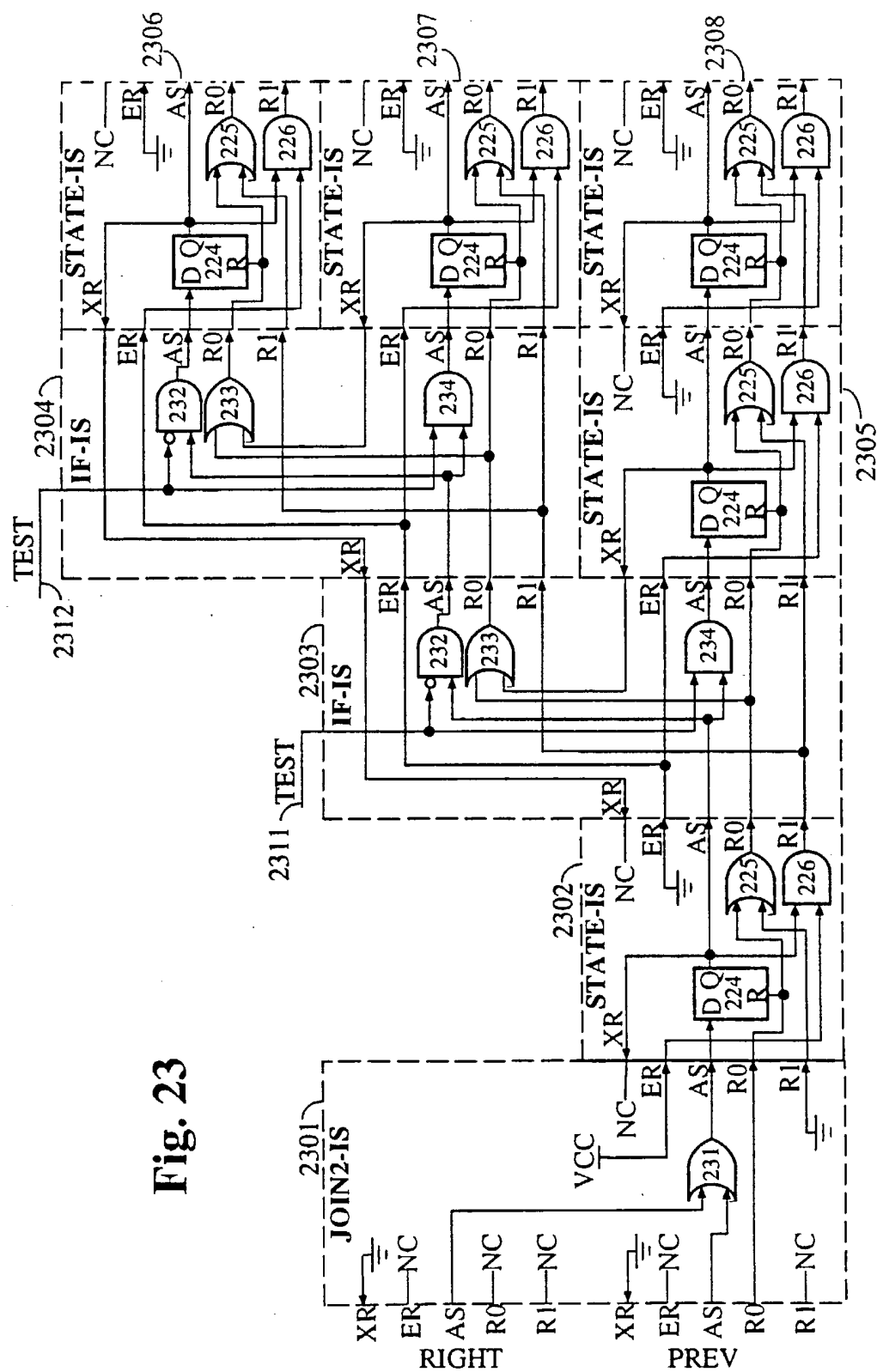


Fig. 23

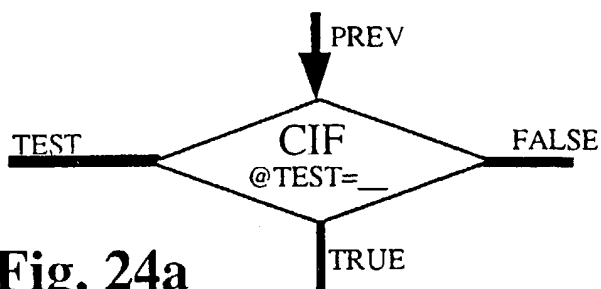


Fig. 24a

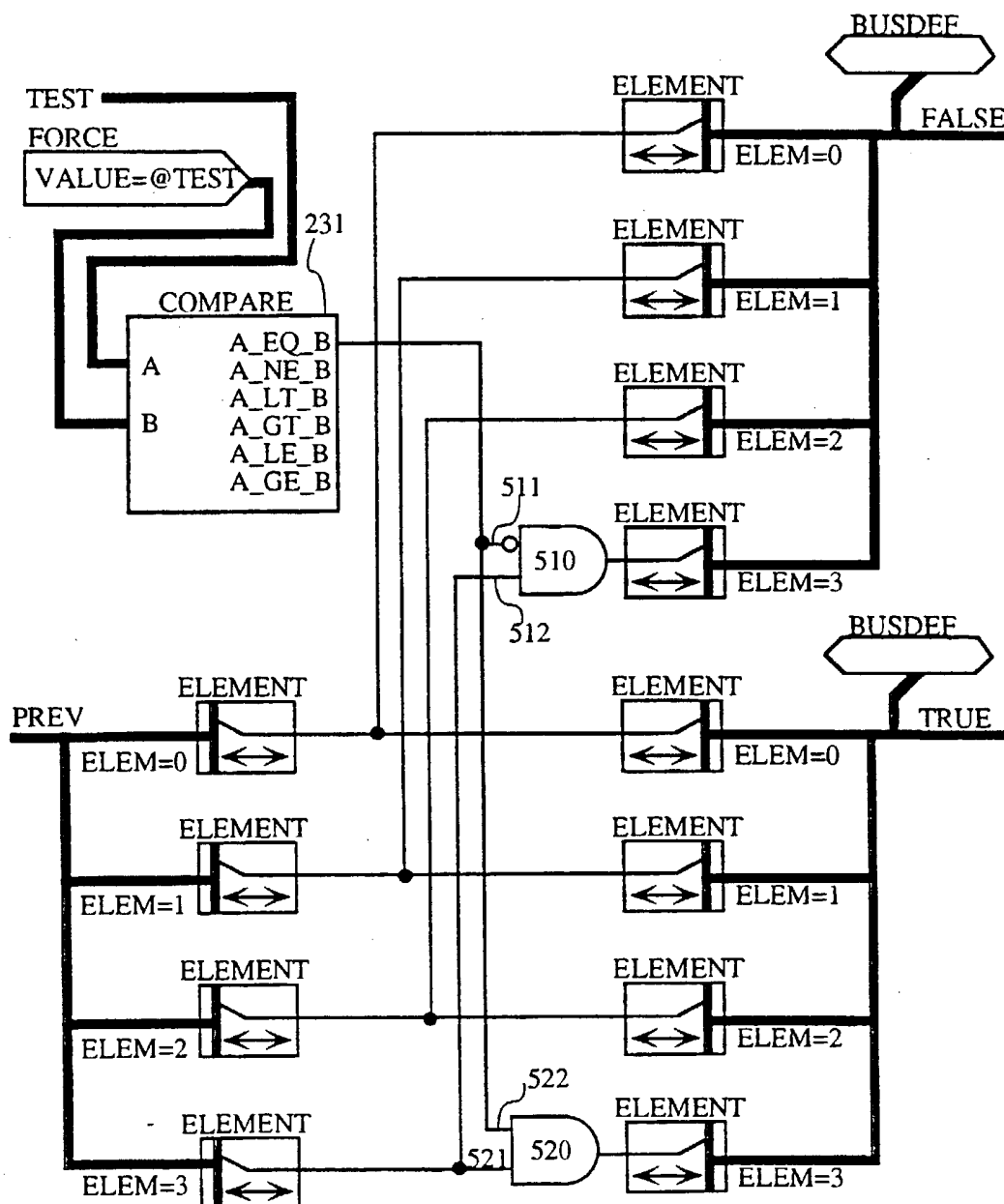


Fig. 24b

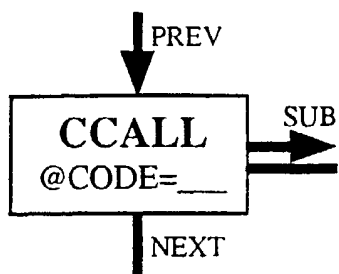


Fig. 25a

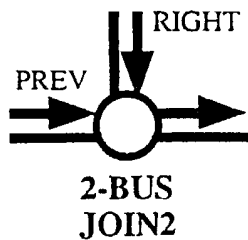


Fig. 25b

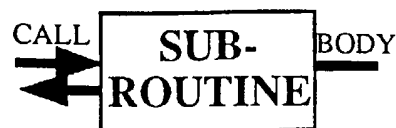


Fig. 25c

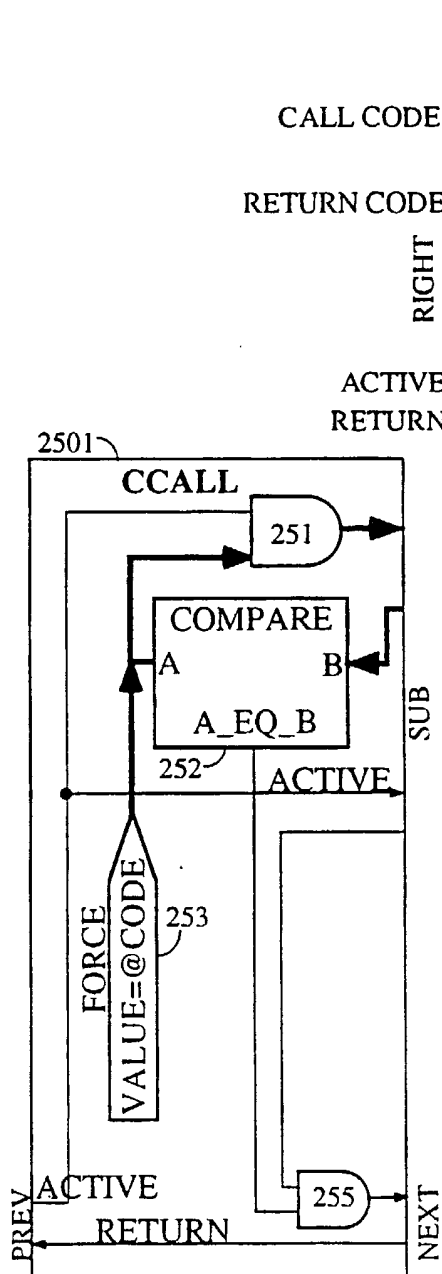


Fig. 25d

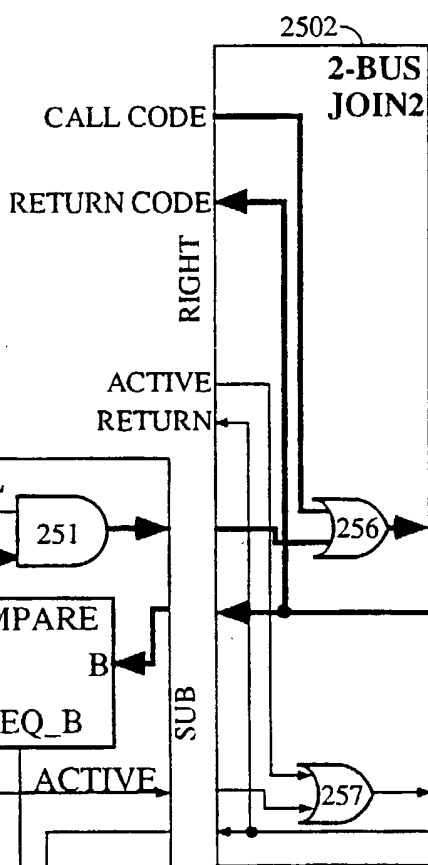


Fig. 25e

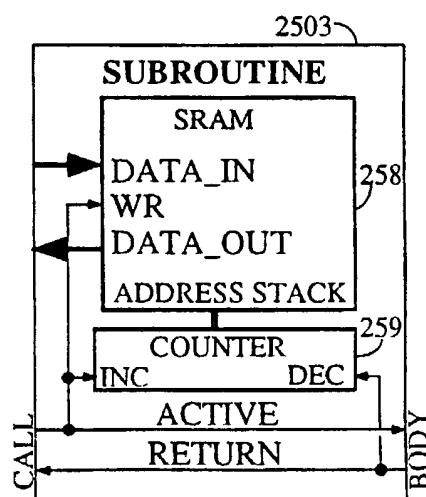


Fig. 25f

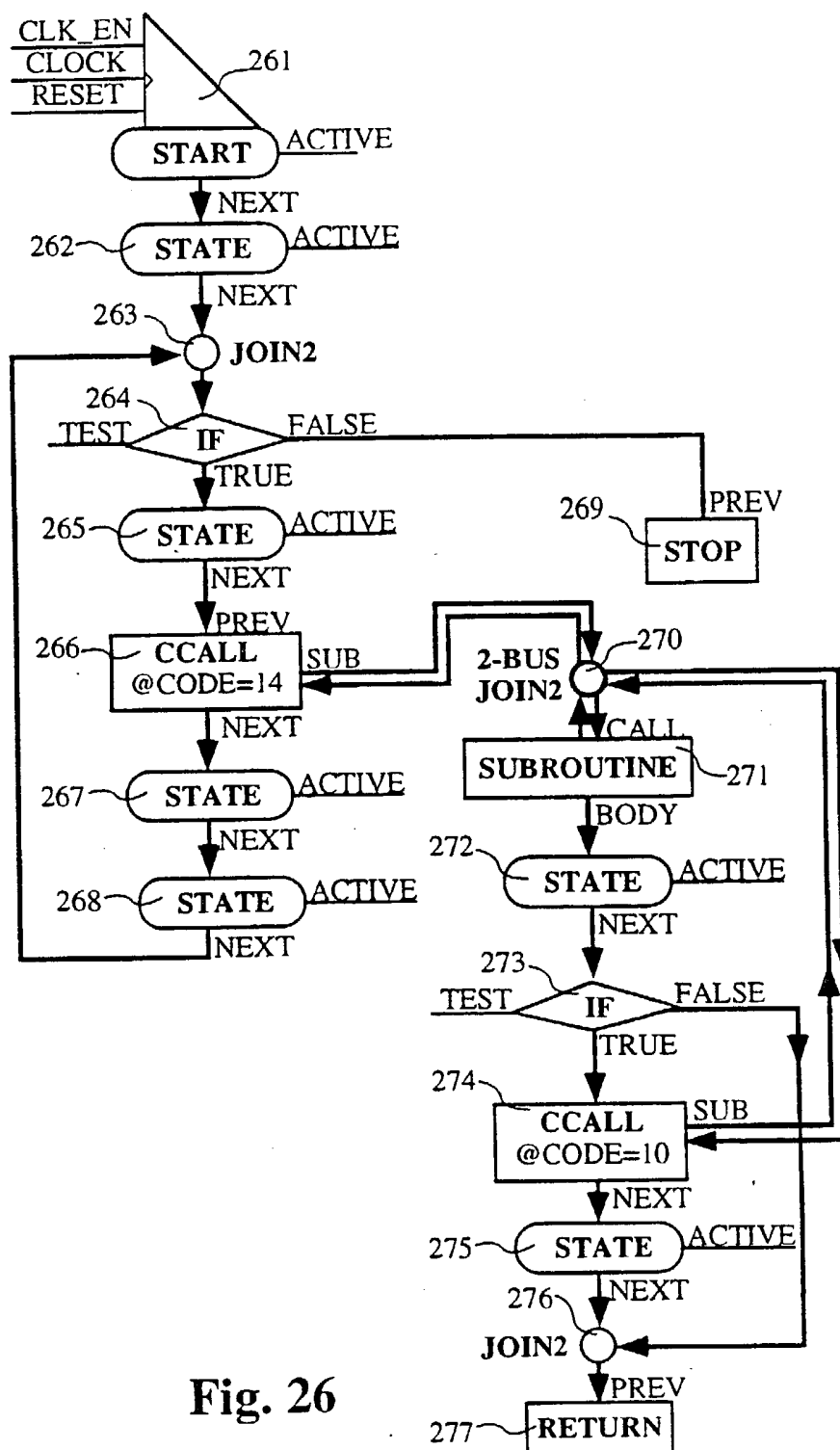


Fig. 26

Fig. 27a

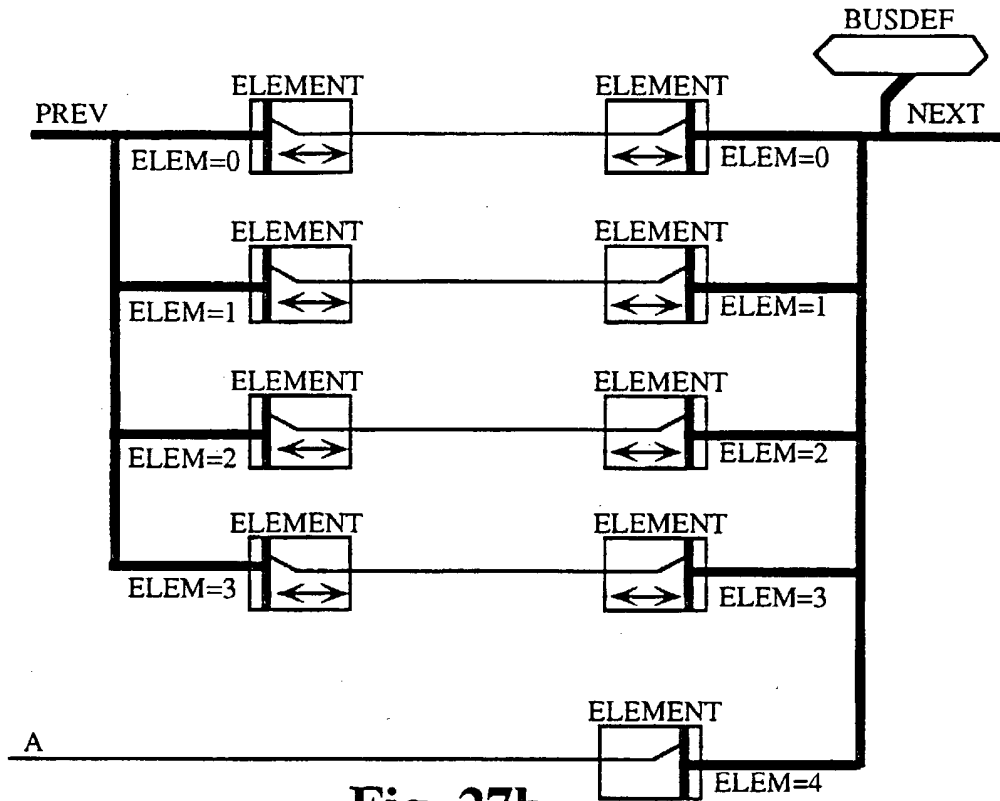
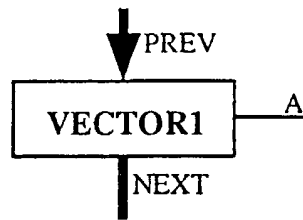


Fig. 27b

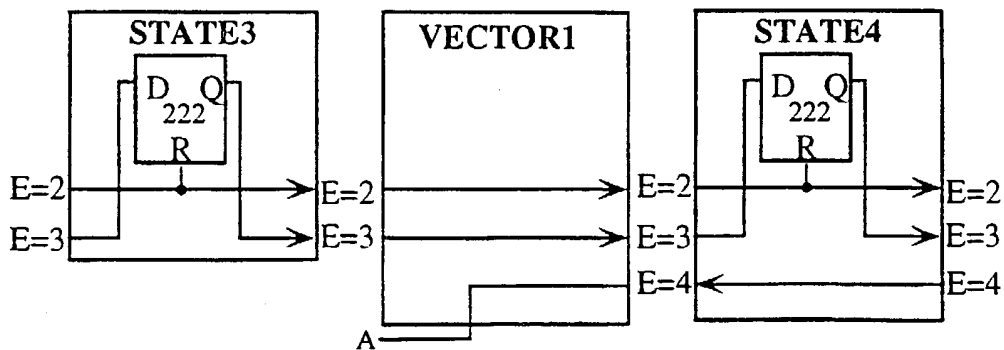


Fig. 27c

Fig. 27d

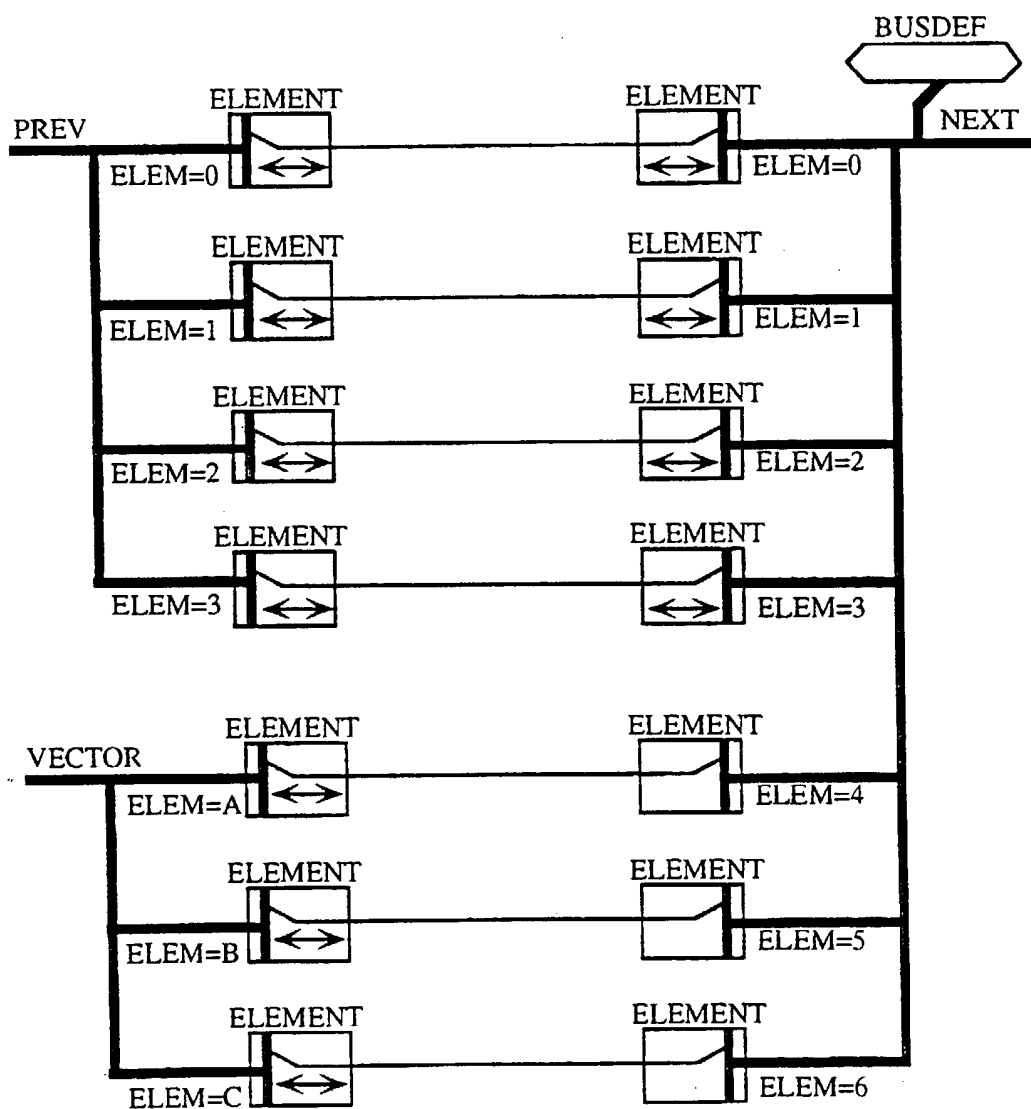
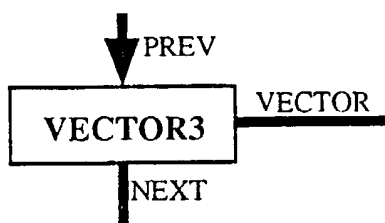


Fig. 27e

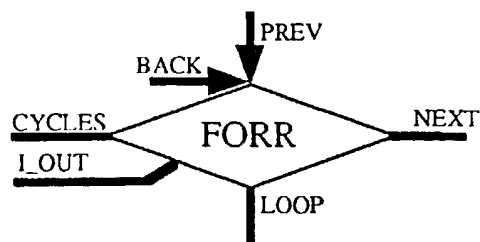


Fig. 28a

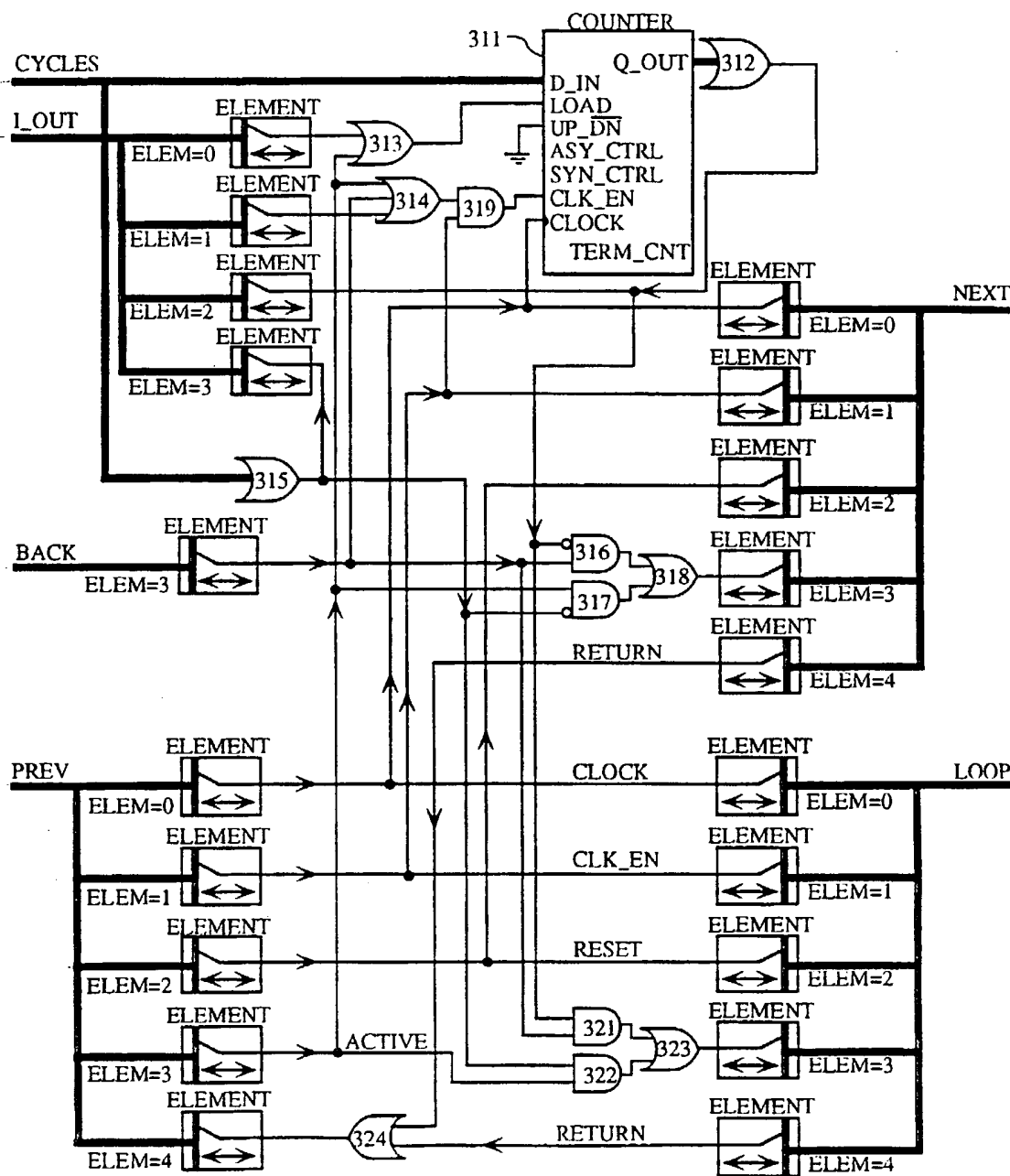


Fig. 28b

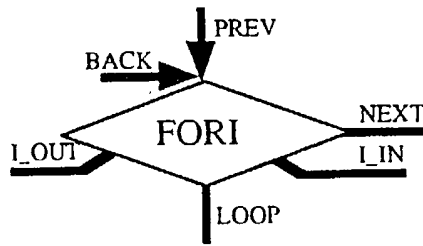


Fig. 29a

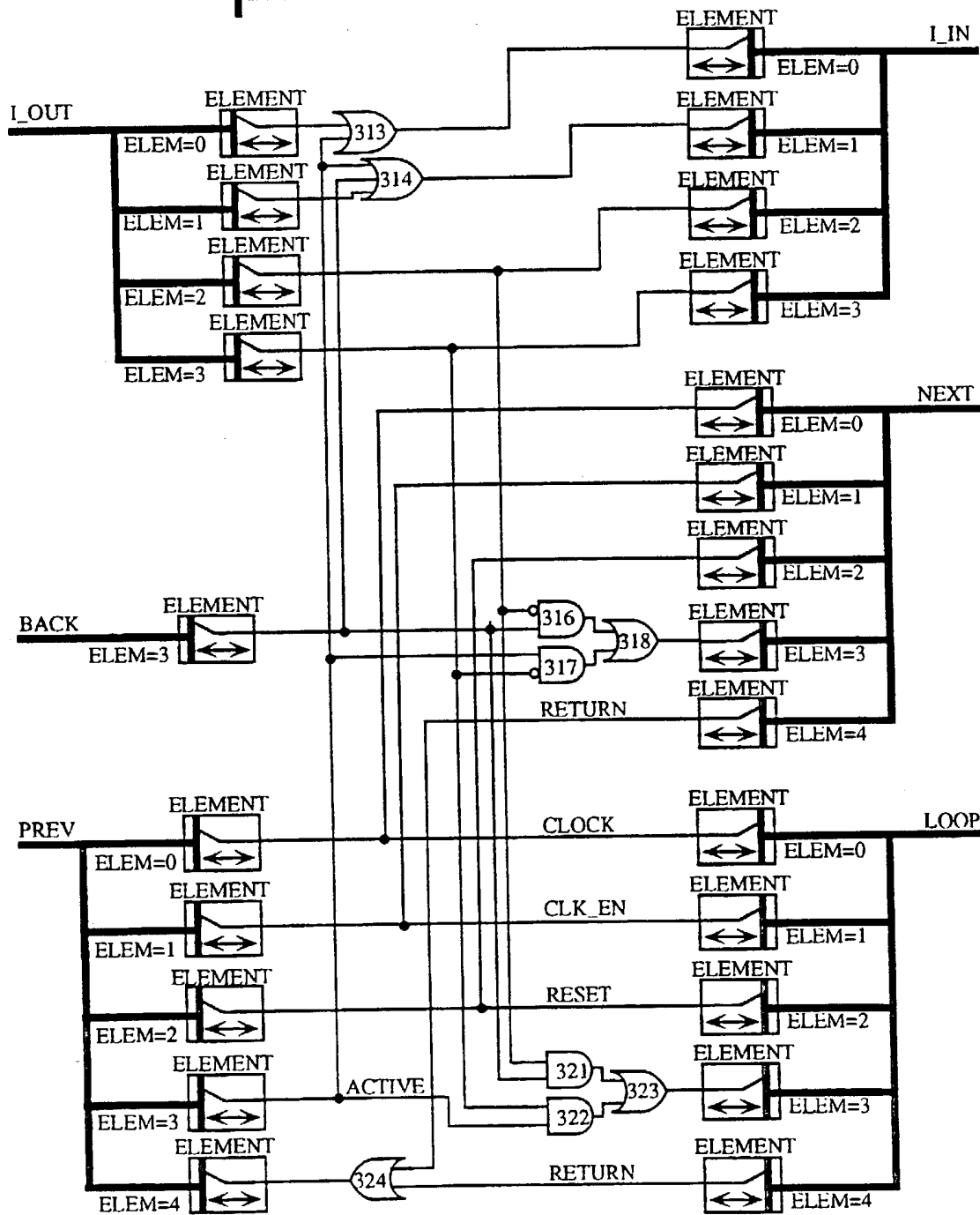


Fig. 29b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/12683

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) : G06F 15/60.

US CL : 364/488, 489, 490.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/488, 489, 490, 578; 395/140, 141, 919, 920, 921.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	View logic - Schematic Design User's Guide, Viewlogic Systems Inc., Version C for use with Workview 4,1, Series I, May 1991, pages 1-1 to 11-7; especially pages 2-1 to 10-8.	1-6, 14-20 ----- 7-13, 21
Y	US, A, 4,827,427 (Hyduke) 02 May 1989. See the entire document.	1-21
X --- Y	US, A, 4,922,432 (Kobayashi et al.) 01 May 1990. See Abstract; Fig.s 1 and 3; and col. 4, line 49 - col. 10, line 37.	7-13, 21 ----- 1-6, 14-20
Y	US, A, 4,967,367 (Piednior) 30 October 1990. See the entire document.	1-21
A	US, A, 4,970,664 (Kaiser et al.) 13 November 1990. See the entire document.	1-21

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	* T	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
* A		document defining the general state of the art which is not considered to be part of particular relevance
* E		earlier document published on or after the international filing date
* L		document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
* O		document referring to an oral disclosure, use, exhibition or other means
* P		document published prior to the international filing date but later than the priority date claimed
	* X	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
	* Y	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
	* &	document member of the same patent family

Date of the actual completion of the international search

07 APRIL 1994

Date of mailing of the international search report

MAY 04 1994

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. NOT APPLICABLE

Authorized officer

Vincent N. Trans

Telephone No. (703) 305-9750

INTERNATIONAL SEARCH REPORT

 International application No.
 PCT/US93/12683

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 5,051,938 (Hyduke) 24 September 1991. See the entire document.	1-21
Y	US, A, 5,084,824 (Lam et al.) 28, January 1992. See the entire document.	1-21
X	US, A, 5,164,911 (Juran et al.) 17 November 1992. See Figs. 2-6.	1-6, 14-20 -----
Y		7-13, 21
X, P	US, A, 5,197,016 (Sugimoto et al.) 23 March 1993. See the entire document.	7-13, 21
Y, P	US, A, 5,210,699 (Harrington) 11 May 1993. See the entire document.	7-13, 21
A, P	US, A, 5,220,512 (Watkins et al.) 15 June 1993. See the entire document.	1-21
A, P	US, A, 5,222,030 (Dangelo et al.) 22 June 1993. See the entire document.	1-21
A, P	US, A, 5,258,919 (Yamanouchi et al.) 02 November 1993. See the entire document.	1-21
Y, E	US, A, 5,278,769 (Bair et al.) 11 January 1994. See the entire document.	1-6, 14-20
X, E	US, A, 5,301,318 (Mittal) 05 April 1994. See the entire document.	1-21